A STUDY OF ELEMENT INTERACTIONS IN THERMOACOUSTIC ENGINES

FINAL REPORT

for

CONTRACT #N00014-93-1-0077

Submitted to:

Office of Naval Research
Program Officer Logan E. Hargrove ONR 331
Ballston Centre Tower One
800 North Quincy Street
Arlington, VA 22217-5660

Submitted by:

Henry E. Bass, Richard Raspet and James R. Belcher
Department of Physics and Astronomy
The University of Mississippi
University, Mississippi 38677

PARGUM REPORT 96-01

May 1996

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		Final Report 01 Oct 92 - 30 Sep 95

4. TITLE AND SUBTITLE

A Study of Element Interactions in Thermoacoustic Engines

5. FUNDING NUMBERS

PE 61153N
G N00014-93-1-0077

6. AUTHOR(S)

Henry E. Bass, Richard Raspet and James R. Belcher

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Department of Physics & Astronomy
University of Mississippi
University, MS 38677

8. PERFORMING ORGANIZATION REPORT NUMBER

PARGUM Report 96-01

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
ONR 331
800 North Quincy Street
Arlington, VA 22217-5660

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

Ph.D. dissertation of James R. Belcher.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release: Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The general performance of a thermoacoustic prime mover or refrigerator is reasonably well understood. There are notable discrepancies between theory and experiment. These discrepancies are typically attributed to non-linear terms not included in the theory. There is evidence, however, that interactions between elements in the thermoacoustic device are at least partially responsible. An experimental investigation of the element interactions in a thermoacoustic prime mover and comparison to theoretical predictions was undertaken in this dissertation to minimize the temperature difference across the stack necessary to achieve onset of self oscillations (Delta T). This was accomplished by varying the position or physical dimensions of the thermoacoustic system or elements to modify the interactions and modifying the working fluid properties through binary mixing.

14. SUBJECT TERMS

Thermoacoustics, Physical Acostics, Refrigeration, Prime Mover

15. NUMBER OF PAGES

222

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

A STUDY OF ELEMENT INTERACTIONS IN THERMOACOUSTIC ENGINES

A DISSERTATION PRESENTED

FOR THE

DOCTOR OF PHILOSOPHY

DEGREE

UNIVERSITY OF MISSISSIPPI

JAMES RICHARD BELCHER

MAY, 1996

To the Graduate Council:

I am submitting herewith a dissertation written by James Richard Belcher entitled "A Study of Element Interactions in Thermoacoustic Engines." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Physics.

Henry E. Bass

Dr. Henry E. Bass, Major Professor
Acting Chair, Dept. of Physics and Astronomy

We have read this dissertation
and recommend its acceptance:

Accepted for the Council:

Dean of The Graduate School

DEDICATION

This dissertation is dedicated to two most
important people I have known
my father
"Smokey"
and
my best friend and love of my life
"Trish".

ACKNOWLEDGMENTS

I would like to thank Dr. Henry Bass and Dr. Richard Raspet for giving me this opportunity and for having the patience and understanding of my unique situation. I would like to thank Dr. Pat Arnott for the long hours and stimulating conversions on thermoacoustics. I would like to thank the other committee member Dr. Bruce Denardo and Dr. Jeffrey Roux, and with special appreciation to Dr. Gordon Baird and Dr. Anthony Atchley. I would like to thank Jay Lightfoot for all the help and friendship one could ask for. I would like to like to express my deepest appreciation to my wife, Trish, and children, Natasha, Brandy, and Travis for their personal sacrifices and support. I would like to thank my father for the motivation and inspiration. I greatly appreciate the Office of Naval Research for providing the funding this research.

ABSTRACT

The general performance of a thermoacoustic prime mover or refrigerator is reasonably well understood. There are notable discrepancies between theory and experiment. These discrepancies are typically attributed to non-linear terms not included in the theory. There is evidence, however, that interactions between elements in the thermoacoustic device are at least partially responsible. An experimental investigation of the element interactions in a thermoacoustic prime mover and comparison to theoretical predictions was undertaken in this dissertation to minimize the temperature difference across the stack necessary to achieve onset of self oscillations (Delta T). This was accomplished by varying the position or physical dimensions of the thermoacoustic system or elements to modify the interactions and modifying the working fluid properties through binary mixing.

Experimental and theoretical results indicate the short stack approximation and stability analysis can be used to optimize the position and pore dimension of the stack respectively for a minimum onset Delta T prime mover. An acoustically stimulated heat flow is observe when the optimize prime mover was modified decreasing the operating temperature difference 20K below onset Delta T.

Binary gas mixtures, monatomic/monatomic or monatomic/polyatomic, can reduce onset Delta T significantly. The components of the binary gas mixture must have a minimum mass ratio of 5 to reduce Delta T. A low onset Delta T is observed

when the polyatomic gas sulfur-hexafluride is used as the working fluid in a thermoacoustic prime mover.

Element losses can be reduced by decreasing the length or surface area interacting with the working fluid reducing onset Delta T. Heat exchanger losses can be reduced without losing the heat transfer effectiveness and convective heat must be in the direction of the stack to further reduce the onset Delta T. A nichrome wire heat exchanger was used to demonstrate the effectiveness of heating the working fluid directly.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Thermoacoustic engines are devices that convert heat energy to acoustic energy or use acoustic energy to facilitate heat transfer. A sufficiently high temperature gradient applied across a porous media (stack), appropriately positioned inside an acoustic resonator, will cause spontaneous oscillations at the frequency corresponding to the resonance of the system. The reverse process is also possible, i.e., placing a stack inside an excited resonator causes a temperature gradient to be generated across the stack, pumping heat from one end of the stack to the other.

A device that converts heat energy into acoustic energy utilizing a temperature gradient across a porous medium is a thermoacoustic prime mover. A device that shuttles heat across the stack by means of a sound wave is referred to as a thermoacoustic heat pump. These two devices can be combined such that the heat pump section uses the acoustic oscillations produced by a prime mover section. This combination is referred to as a heat driven thermoacoustic refrigerator.

The earliest example of thermoacoustics was the sound emitted during glass blowing; when a cool cylindrical stem was attached to a hot bulb the

system sometimes would emit sound. Sondhauss[1] was the earliest pioneer in thermoacoustics with his investigation of the frequency emitted as a function of the size of the bulb.

Baron Rayleigh's [2] qualitative explanation of this phenomenon, which has now become known as Sondhauss oscillation, is an eloquent explication of thermoacoustics. "In almost all cases where heat is communicated to a body expansion ensues, and this expansion may be made to do mechanical work. If the phases of the forces thus operative be favorable, a vibration may be maintained .... For sake of simplicity, a tube, hot at the closed end, may be considered. At a quarter of a period before the phase of greatest condensation... the air is moving inwards, i.e., towards the closed end, and therefore is passing from colder to hotter parts of the tube, ... But in fact adjustment of temperature takes time, and thus the temperature of the air deviates from that of the neighboring parts of the tube, inclining towards the temperature of that part of the tube from which the air has just come. From this it follows that at the phase of greatest condensation heat is received by the air, and at the phase of greatest rarefaction heat is given up from it and thus there is a tendency to maintain the vibrations."

Kirchhoff's theory[3] on acoustic wave damping in tubes due to frictional forces acting within the boundary layer was published in 1868. He included

the previously neglected heat transfer to the wall of the tube. This was the starting point for thermoacoustic theories, but would be ignored for almost a century.

The study of cryogenics led indirectly to the next advancement in thermoacoustics. A temperature difference generated along the length of a gas filled tube, in excess of 370K, was observed to cause high amplitude acoustic oscillations. Taconis' [4] quantitative explanation of this phenomenon was very similar to Baron Rayleigh's discussion of Sondhauss oscillations. Kramers [5] generalized Kirchhoff's equations for strong temperature gradients along the tube wall, but theoretical models were not in agreement with experiment. Shortly thereafter, Rott and his coworkers' [6-11] began a series of articles to theoretically explain Taconis oscillations. Rott *et al.* [6-11] using the theories of Kirchoff and Kramers, with the assumption that the radial gradient of the acoustic pressure can be neglected, developed a linearized thermoacoustic theory for cylindrical tubes. This was experimentally verified by Yakazi *et al.*[12]

The Sondhauss tube and Taconis oscillator are examples of thermoacoustic prime movers where there are only two elements present, the working fluid and the resonator. Merkli and Thomann [13] presented experimental results showing heating and cooling of a resonator's walls in a piston driven tube. These results are some of the earliest examples of a dual

element thermoacoustic heat pump. The heating occurred at the closed end of the resonator where pressure is maximum and cooling occurred at the center of the tube where velocity is maximum.

Wheatley *et al.* [14] extended Rott's work to include parallel plate structures (the thermoacoustic couple now referred to as the stack) inside an externally excited acoustic resonator. Their experiment and analysis showed cooling at one end and heating at the other end of the stack. The shuttling of heat was attributed to an acoustically stimulated entropy flow into the end of the stack closest to the pressure antinode and out of the end of the stack closest to the pressure node. The efficiency of these devices depends on geometry, configuration, and gas properties rather than temperature difference.

Hofler [15] constructed a thermoacoustic prime mover as a demonstration for his doctoral candidacy exam. The device uses a 8.54 cm long cylindrically shaped stack of 0.38 mm thick fiber-glass plates spaced 1 mm apart. This device is similar to the Sondhauss tube or Taconis oscillator, but the efficiency is increased due to the stack.

Swift [16] gives a complete historical review of experimental thermoacoustics from the Sondhauss tube and Taconis oscillator through the Hofler prime mover. He also reviews thermodynamics from the basic principles of heat engines to the advanced principles of thermoacoustics

presented by Rott and expanded by Wheatley. Swift's discussion and analysis of the thermoacoustic prime mover and its components leaves the reader with an intuitive understanding of thermoacoustics.

Atchley *et al.* [17] measured the frequency response of a cylindrical tube with a stack about 1/8 wavelength from one end as a function of the mean pressure and temperature gradient across the stack. The quality factor was computed by a fitting routine applied to the data. A counter-propagating, plane wave model commonly used in porous media theories was used to explain the result. The agreement was very good except at low ambient pressures and no temperature gradient cases.

Atchley [18] later used Swift's short stack approximation that the acoustic pressure and velocity can be described in terms of standing waves throughout the resonator, to analyze a thermoacoustic prime mover before onset of self oscillations. The pressure and velocity were written in terms of transcendental functions and the stored energy was calculated leading directly to the quality factor. Comparison with experimental values shows good agreement at high ambient pressures.

Arnott *et al.* [19] generalized thermoacoustics to include different pore geometries, including the parallel plates and circular pores originally considered by Rott, by approaching thermoacoustics from capillary tube

porous media theory. By defining the single pore transport function (F), which represents the functional form of the transverse variation of the longitudinal particle velocity, the first-order acoustical field quantities and the second order energy flux can be evaluated for any pore shape once F is known.

Raspet *et al.* [20] presented a theoretical analysis of traveling waves applied to thermoacoustics. The work done on the propagating wave is calculated in the invisid boundary layer approximation. Kordomenos *et al.* [21] verified this theory experimentally for a traveling wave tube termination.

Arnott *et al.* [22] have recently completed the theoretical analysis of a radial thermoacoustic prime mover. Impedance and pressure translation equations are computed for the radial resonator and heat exchangers. The first order differential equations for pressure and impedance are presented for temperature gradients across the stack and are used to calculate the heat and work flows. The comparison between plane wave and radial thermoacoustic devices are investigated under different conditions. Lightfoot [23] is experimentally investigating the possibility of a radial thermoacoustic device as part of his dissertation.

Numerical techniques have been developed using Runge-Kutta integration to analyze different systems. Ward *et al.* [24] integrate the acoustical

equations to calculate the field equation in the different elements. Arnott [25] uses translation theories for impedance and pressure to calculate the quantities.

The general performance of themoacoustic prime movers and refrigerators are reasonably well understood and documented. There are notable discrepancies between theory and experiment. These discrepancies are typically attributed to non-linear terms not included in the theory. There is evidence, however, that interactions between elements in the prime mover are at least partially responsible for the difference. Additional elements are required in heat driven refrigerators leading to more interactions.

This dissertation describes research performed to investigate interactions between the elements of a thermoacoustic prime mover. The experiments are designed to measure the temperature difference required to reach onset of self oscillation under different conditions. The reduction in onset temperature is necessary to optimize a heat driven thermoacoustic refrigerator to produce cooling from waste heat.

The experimental results are compared to theoretical predictions derived from the theory[19] presented in Chapter 2. The construction and optimization[26] of a helium filled prime mover is discussed in Chapter 3. The working fluid interactions with the different elements are investigated in Chapters 4. The heat exchangers are investigate in Chapter 5. Chapter 6

includes the interactions caused by the addition of a heat pump section. Conclusions in Chapter 7 compares the experimental results to the linear theory to reveal needed areas of improvement. Included are suggestion on optimizing a heat driven thermoacoustic refrigerator designed to provide cooling from a waste heat source.

Chapter 2

Review of Thermoacoustic Theory

2.1 Introduction

A common approach for the theory of sound in porous media is to envision the medium as a collection of capillary tubes. The equations and boundary conditions used in porous media modeling and in thermoacoustics are nearly identical (thermoacoustics has an extra term proportional to the ambient temperature gradient).

Arnott *et al.* [19] generalized thermoacoustics using the porous media approach of defining a thermoviscous function ($F(\lambda)$) to include various pore geometries. This theory is reviewed here as the foundation for the research presented. The use of a fourth order Runge Kutta integration to calculate the impedance and pressure across the stack was developed by Arnott. The numerical routines are presented in appendix A.

2.2 Fluid Field Equations

The transverse coordinates are taken to be x and y, and the longitudinal coordinate z. The fluid quantities in a pore to the first order are:

$$P(z) = P_0 + P_1(z)\exp(-i\omega t), \qquad 2.1$$

$$v(x,y,z,t) = \left[\vec{v}_\tau(x,y,z) + v_z(x,y,z)\hat{z}\right]\exp(-i\omega t), \qquad 2.2$$

$$T(x,y,z,t) = T_0(z) + T_1(x,y,z)\exp(-i\omega t), \qquad 2.3$$

$$s(x,y,z,t) = s_0(z) + s_1(x,y,z)\exp(-i\omega t), \qquad 2.4$$

and

$$\rho(x,y,z,t) = \rho_0(z) + \rho_1(x,y,z)\exp(-i\omega t). \qquad 2.5$$

These are the equations for pressure, P, particle velocity, v, temperature, T, entropy, s, and density, ρ. Ambient values are represented with subscript 0, acoustical values with subscript 1, longnitudinal values with subscript z, and subscript τ represents the transverse direction. The ambient values of velocity, temperature, entropy, and density are all functions of the longitudinal coordinate. We are assuming exp(-iωt) time dependence.

The Navier-Stokes equations to the first order are:

$$-i\omega\rho_0 v_z(x,y,z) = -\frac{dP_1(z)}{dz} + \eta\nabla_\tau^2 v_z(x,y,z), \qquad 2.6$$

$$-i\omega\rho_1(x,y,z) + \frac{\partial\left(\rho_0(z)v_z(x,y,z)\right)}{\partial z} + \rho_0(z)\nabla_\tau\cdot\vec{v}_\tau(x,y,z) = 0, \qquad 2.7$$

$$\rho_1(x,y,z) = -\rho_0(z)\beta T_1(x,y,z) + \left(\frac{\gamma}{c^2}\right)P_1(z), \qquad 2.8$$

$$s_1(x,y,z) = \left(\frac{c_p}{T_0}\right)T_1(x,y,z) - \left(\frac{\beta}{\rho_0}\right)P_1(z), \qquad 2.9$$

and

$$-i\omega\rho_0(z)c_p T_1(x,y,z) + \rho_0(z)c_p v_z(x,y,z)T_{0z} = -i\omega\beta T_0 P_1(z) + \kappa\nabla_\tau^2 T_1(x,y,z), \qquad 2.10$$

where the transverse gradient and Laplacian operators are defined by

$\nabla_\tau = \frac{\partial}{\partial x}\hat{x} + \frac{\partial}{\partial y}\hat{y}$, $\nabla_\tau^2 = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)$, c_p is the isobaric heat capacity per unit mass, γ is the ratio of specific heats, ω is the angular frequency, β is the thermal expansion

coefficient, κ is thermal diffusitivity, η is the coefficient of viscosity and $T_{0z} = \frac{\partial T_z}{\partial z}$.

These equations represent the longitudinal component of the equation of motion, continuity equation, equation of state for density and entropy, and the heat transfer equation respectively.

2.3 Transverse Velocity

The equation of motion, Equation 2.6, can be rewritten as

$$v_z(x,y,z) = \frac{F(x,y;\lambda)}{i\omega\rho_0}\frac{dP_1(z)}{dz} \qquad 2.11$$

where $F(x,y,\lambda)$ satisfies

$$F(x,y,\lambda) + \left(\frac{R^2}{i\lambda^2}\right)\nabla_t^2 F(x,y;\lambda) = 1 \qquad 2.12$$

subject to the boundary condition that $F(x,y,\lambda)$ is zero at the pore walls, which is equivalent to the no slip boundary condition for the particle velocity.

The equation of motion for the fluid averaged over the pore cross section can then be expressed as

$$v_z(z) = \frac{1}{i\omega\tilde{\rho}(z;\lambda)}\frac{dP_1(z)}{dz}, \qquad 2.13$$

where $\tilde{\rho}(z;\lambda) = \frac{\rho_0(z)}{F(\lambda)}$ is the complex density. This is the apparent dynamic density of the fluid.

2.4 Transverse Temperature

The excess temperature in the fluid, due to the compression and rarefaction, is given by Equation 2.10. For an ideal gas we can use the thermodynamic relationship $\frac{T_0\beta^2}{c_p} = \frac{(\gamma-1)}{c^2}$, where c is the sound speed and rewrite Equation 2.10 as

$$T_1(x,y,z) + \left(\frac{R^2}{i\lambda_\tau^2}\right)\nabla_\tau^2 T_1(x,y,z) = \frac{\gamma-1}{c^2\rho_0\beta}P_1(z) - \frac{T_{0z}}{\rho_0\omega^2}F(x,y;\lambda)\frac{dP_1(z)}{dz} \qquad 2.14$$

after application of the boundary condition $T_1(x,y,z)=0$ for both x and y boundaries. The excess temperature can then be written as

$$T_1(x,y,z) = \frac{\gamma - 1}{c^2 \rho_0 \beta} P_1(z) F(x,y;\lambda_T) - \frac{T_{0z}}{\rho_0 \omega^2} \frac{F(x,y,\lambda_T) - N_{pr} F(x,y;\lambda)}{1 - N_{pr}} \frac{dP_1(z)}{dz} \quad 2.15$$

The equation of state for the density in the pore is represented by Equation 2.8 and can be rewritten using Equation 2.15 as

$$\rho_1(x,y,z) = \frac{1}{c^2}\left[\gamma - (\gamma - 1)F(x,y;\lambda_T)\right] P_1(z) + \frac{\beta T_{0z}}{\omega^2} \frac{F(x,y;\lambda_T) - N_{pr} F(x,y;\lambda)}{1 - N_{pr}} \frac{dP_1(z)}{dz} \quad 2.16$$

Averaging over the cross sectional area of the pore, the density can be represented by

$$\rho_1(z) = \left[\gamma - (\gamma - 1)F(\lambda_T)\right] \frac{P_1(z)}{c^2} + \frac{\beta T_{0z}}{\omega^2} \frac{F(\lambda_T) - N_{pr} F(\lambda)}{1 - N_{pr}} \frac{dP_1(z)}{dz}. \quad 2.17$$

2.5 Differential Equation for Pressure

The continuity equation, equation of motion, and the equation of state for the density can be combined to yield an equation for the pressure in a pore. Averaging the continuity equation over the pore area yields

$$-i\omega\rho_1(z)+\frac{d}{dz}\left(\rho_0(z)v_z(z)\right)=0 \qquad 2.18$$

or

$$-i\omega\rho_1(z)+\rho_0(z)\frac{d}{dz}v_z(z)-\beta T_{0z}v_z(z)=0. \qquad 2.19$$

using the equation of state for an ideal gas. Using Equation 2.11 in Equation 2.19 the continuity equation can be written as

$$-i\omega\rho_1(z)+\frac{\rho_0(z)}{i\omega}\frac{d}{dz}\left(\frac{F(\lambda)}{\rho_0}\frac{dP_1(z)}{dz}\right)-\beta T_{0z}\frac{F(\lambda)}{\rho_0}\frac{dP_1(z)}{dz}=0 \qquad 2.20$$

Using Equation 2.17 for $\rho_1(z)$ along with Equation 2.20 multiplied by $i\omega/F(\lambda)$ the equation for pressure can be written as

$$\frac{\rho_0}{F(\lambda)}\frac{d}{dz}\left(\frac{F(\lambda)}{\rho_0}\frac{dP_1(z)}{dz}\right)+2\alpha(\lambda,\lambda_T)\frac{dP_1(z)}{dz}+k(\lambda,\lambda_T)^2P_1(z)=0 \qquad 2.21$$

where

$$\alpha(\lambda,\lambda_T)=\frac{\beta T_{0z}}{2}\left(\frac{F(\lambda_T)/F(\lambda)-1}{1-N_{pr}}\right) \qquad 2.22$$

and

$$k(\lambda,\lambda_T)^2=\frac{\omega^2}{c^2}\frac{1}{F(\lambda)}\left[\gamma-(\gamma-1)F(\lambda_T)\right]. \qquad 2.23$$

In the absence of a temperature gradient $T_{0z}=0$ so $\alpha(\lambda,\lambda_T)=0$. The complex wave number in the pore is then given by $\pm k$, which is the usual form found in porous media models.

2.6 Heat and Work flow

The time averaged energy flow to the second order is

$$\overline{\dot{H}}_2(z)=\overline{\dot{Q}}_2(z)+\overline{\dot{W}}_2(z)-\overline{\dot{Q}}_{loss}(z), \qquad 2.24$$

where time-averaged heat flow due to hydrodynamic transport is

$$\overline{\dot{Q}}_2(z) = \frac{\Omega A_{res}}{2} \mathrm{Re}\frac{1}{A} \int_A [\rho_0 c_p v_z(x,y,z) T_1^*(x,y,z) - \beta T_0 v_z(x,y,z) P_1^*(z)] dxdy. \quad 2.25$$

The heat flow due to thermal conduction down the temperature gradient is

$$\overline{\dot{Q}}_{loss}(z) = \Omega A_{res} \kappa_{gas} T_{0z} + (1-\Omega) A_{res} \kappa_{stack} T_{0z} , \quad 2.26$$

and the time averaged work flow is

$$\overline{\dot{W}}_2(z) = \frac{\Omega A_{res}}{2} \mathrm{Re}\frac{1}{A} \int_A [v_z(x,y,z) P_1^*(z)] dxdy . \quad 2.27$$

Substituting for the velocity and temperature and integrating, the heat and work flows can be written as

$$\overline{\dot{Q}}_2(z) = -\frac{\Omega A_{res}}{2} \beta T_0 \left\{ \mathrm{Im}\left[\frac{dP_1(z)}{dz} \frac{P_1^*(z)}{\rho_0 \omega} \frac{F^*(\lambda_T) - F(\lambda)}{1 + N_{pr}} \right] - \frac{T_{0z}}{\beta T_0} \frac{c_p}{\rho_0 \omega^3} \left| \frac{dP_1(z)}{dz} \right|^2 \frac{\mathrm{Im}\left(F^*(\lambda_T) + N_{pr} F(\lambda)\right)}{1 - N_{pr}^2} \right\}, \quad 2.28$$

and

$$\overline{\dot{W}_2(z)} = \frac{\Omega A_{res}}{2} \mathrm{Im}\left(\frac{\frac{dP_1(z)}{dz} P_1^*(z)}{\rho_0 \omega} F(\lambda) \right). \qquad 2.29$$

Equations 2.28 and 2.29 are general expressions for heat and work flows with the functional form of $F(\lambda)$ dependent on the particular pore geometry.

2.7 Short stack approximation

The stack is assumed to be short enough that the empty tube standing wave is unaffected. The pressure, $P_1(z)$, and particle velocity, $v_z(z)$, of the thermoacoustic engine are represented by

$$P_1^s(z) = P_1(0)\cos k_0 z \qquad 2.30$$

and

$$v_z^s(z) = \frac{P_1(0)}{\Omega \rho_0 c}\sin k_0 z \qquad 2.31$$

where

$$v_z(z) = iv_z^s(z), \tag{2.32}$$

the wave number in the empty tube is $k_0=\omega/c$ and c is the sound speed in free space, Ω is the stack porosity, and z is the distance from the resonator end closest to the stack. The volume of gas in the stack is $V_G=A_{res}\Omega d$ where d is its length and A_{res} is its cross-sectional area of the resonator.

The heat flow can then be written as

$$\begin{aligned}\overline{\dot{Q}}_2(z) = &-\frac{\Omega A_{res} P_1^s(z) v_z^s(z)}{2}\beta T_0 \frac{\mathrm{Im}\left(F^*(\lambda_T)/F(\lambda)\right)}{1+N_{pr}} \\ &+\frac{\rho_0 c_p \Omega A_{res} v_z^{s2}(z)}{2}\frac{\mathrm{Im}\left(F^*(\lambda_T)+N_{pr}F(\lambda)\right)}{\left(1-N_{pr}\right)^2|F(\lambda)|^2}\xi_z^s(z)\frac{T_H-T_C}{d},\end{aligned} \tag{2.33}$$

where $\xi^s_z(z)$ is the particle dissplacement The first term is the conversion of acoustic power to heat and the second term is heat transported due to the temperature gradient. The work flow can be written as

$$\overline{\dot{W}_2}(z) = -\omega \frac{V_G P_1^{s2}(z)}{2\rho_0 c^2}(\gamma - 1)F^*(\lambda_T)$$
$$- \frac{\rho_0 V_G v_z^{s2}(z)}{2} \frac{\mathrm{Im}F^*(\lambda_T)}{|F(\lambda)|^2} \qquad 2.34$$
$$+ \frac{\Omega A_{res}}{2} P_1^s(z) v_z^s(z) \beta (T_H - T_C) \frac{\mathrm{Im}\left[\frac{F^*(\lambda_T)}{F(\lambda)}\right]}{1 - N_{pr}}.$$

The first and second terms are always negative and are the dissipation of potential and kinetic energy per unit time due to thermal and viscous processes in the stack.

2.8 Thermoviscous Function

Thermoacoustic gain and thermoviscous dissipation of sound in porous materials may be described theoretically by a thermoviscous dissipation function.[19] Viscous effects are then expressed in terms of $F(\lambda)$, where $\lambda = R\left(\omega\rho_m/\eta\right)^{1/2}$ is the shear wave number and thermal effects are expressed as $F(\lambda_T)$, where $\lambda_T = R\left(\omega\rho_m c_p/k\right)^{1/2}$ is the thermal disturbance number. R is twice the hydraulic radius of the pore, ω is the angular frequency, ρ_m the ambient density, c_p the specific heat at constant pressure, η the coefficient of viscosity, and k is the thermal conductivity.

The thermoviscous dissipation function $F(\lambda)$ quantifies the exchange of momentum and heat between fluid and nearby solid walls through diffusion processes involving both fluid viscosity and thermal diffusivity and can depend strongly on the geometry of the interaction regime between fluid and solid. The thermoviscous dissipation functions, $F(\lambda)$, and transverse pore dimensions, R, for various pore geometries are presented in Table 2.1.

2.9 Energy losses

The thermal and viscous losses for any additional element can be derived from Equations 2.28 and 2.29. The thermoviscous dissipation function for the pore characteristic of the element is substituted for $F(\lambda)$ and the element is assumed to be isothermal. Equations 2.28 and 2.29, then reduce to

$$\overline{\dot{Q}}_2(z) = -\frac{\Omega A_{res}}{2}\beta T_0 \left\{ \mathrm{Im}\left[\frac{\frac{dP_1(z)}{dz} P_1^*(z)}{\rho_0 \omega} \frac{F^*(\lambda_T) - F(\lambda)}{1 + N_{pr}} \right] \right\}, \qquad 2.35$$

and

Stack Type	Dimensions	$F(\lambda_j)$	Transverse pore dimension
Parallel plate[6,16,19]	plate separation a	$1-\frac{2}{\lambda_j\sqrt{-i}}\tanh\left(\frac{\lambda_j\sqrt{i}}{2}\right)$	R=2a
Cylindrical pore[6,16,19]	Radius a	$1-\frac{2}{\lambda_j\sqrt{-i}}J_1\left(\frac{\lambda_j\sqrt{i}}{2}\right)\Big/Jo\left(\frac{\lambda_j\sqrt{i}}{2}\right)$	R=a
Rectangular pore[19,34]	Dimensions a and b	$\frac{64}{\pi^4}\sum_{\substack{m,n\\odd}}\frac{1}{m^2n^2Y_{mn}(\lambda_j)}$ where $Y_{mn}=\left(1+i\frac{\pi^2(b^2m^2+a^2n^2)}{(a+b)^2}\right)$	$R=\frac{2ab}{a+b}$
Wide tube[35]	Radius = a	$1-2i/\lambda_j$	R = radius
Triangular pore[36]	Sides of length a	$1-\frac{2}{\lambda_j\sqrt{-i}}\coth\left(\frac{3\lambda_j\sqrt{i}}{2}\right)+\frac{4i}{3\lambda_j^2}$	$R=\frac{a}{\sqrt{3}}$

Table 2.1

$$\overline{\dot{W}_2(z)} = \frac{\Omega A_{res}}{2} \mathrm{Im}\left(\frac{\frac{dP_1(z)}{dz} P_1^*(z)}{\rho_0 \omega} F(\lambda) \right). \qquad 2.36$$

The standing wave assumptions used in Section 2.6 can be used to reduce these equations further.

2.10 Conversions to Other Notation

The conversion from porous media notation to the notation used by Swift and others can be accomplished by substituting $F(\lambda) = 1 - f_\nu^*$ and $F(\lambda_T) = 1 - f_\kappa^*$ and noting that $\delta_\nu = 2^{1/2} R/\lambda$ and $\delta_\kappa = 2^{1/2} R/\lambda_T$. The time dependence used in the porous media approach is $\exp(-i\omega t)$, which must be changed to $\exp(+i\omega t)$.

Chapter 3

Prime Mover

3.1 Description

Prime movers are devices that convert thermal energy into acoustical energy. The basic thermoacoustic prime mover can be separated into six individual elements. These are the hot end of the resonator, working fluid (gas), hot heat exchanger, stack, ambient heat exchanger, and the ambient end resonator. Figure 3.1 shows the relative positions of each element in the construction of a prime mover.

A temperature gradient is imposed across the porous media (stack) by means of the heat exchangers at each end. The heat exchangers are typically parallel copper plates (fins) heated or cooled externally. In the prime mover used for these experiments, electrical rod heaters supply heat to the ends of the individual heat exchanger fins. The heat is conducted along their lengths into the resonator. The gas, along the lengths of the individual fins, receives heat via conduction and convection. In the cold heat exchanger, heat is conducted from the working fluid into the individual fins and is conducted along their length toward their ends. Cooling fluid is circulated along the ends of the fins to carry away the excess heat.

The rod heaters and cooling fluid are thermal contact with a 1.90cm thick copper block. The copper block acts as a heat reservoir for the hot heat exchangers

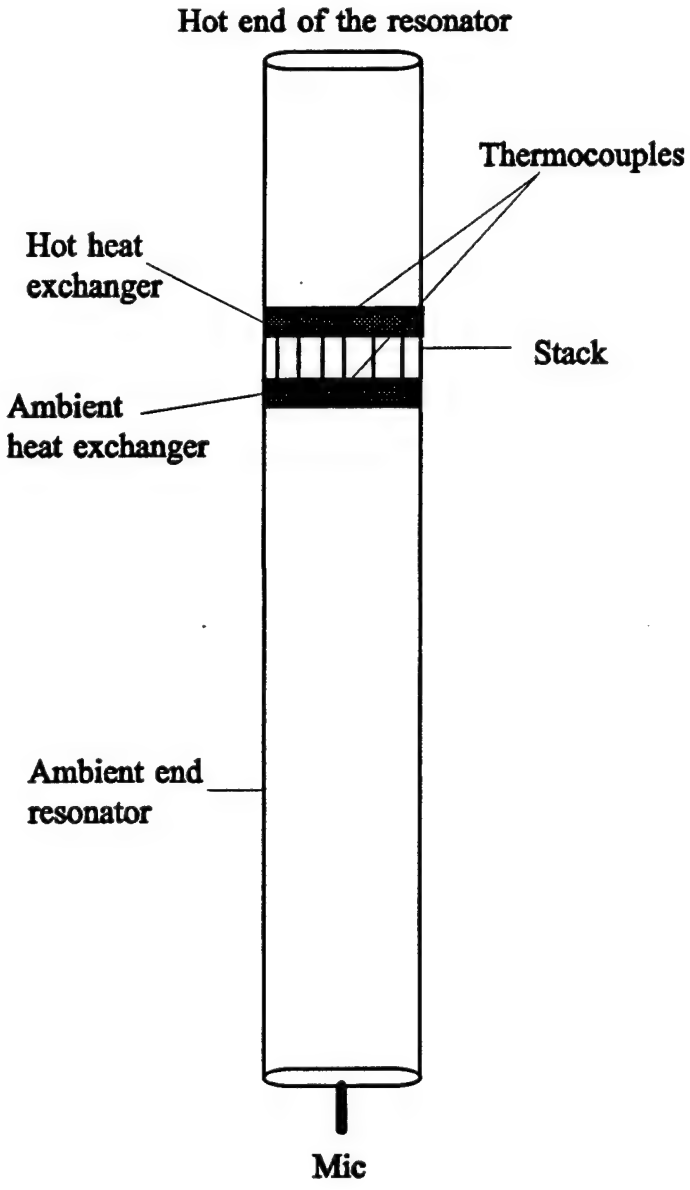

Figure 3.1. The relative positions of the elements in a thermoacoustic prime mover.

and a heat sink for the cold heat exchangers eliminating large fluctuations in temperature over short periods of time.

The stack localizes the thermodynamic interactions and increases the area of interaction with the working fluid. The stack is composed of a porous medium with symmetrical pores that span its entire length. The heat supplied and extracted at the pore ends generate a temperature gradient along its length (assumed to be linear). Random acoustic fluctuations in the resonator are amplified across the stack but decay due to attenuation until the temperature gradient is large enough to balance the attenuation and maintain the oscillations.

Onset of self oscillation is achieved when the amount of thermal energy converted into acoustical energy is greater than the amount being dissipated by thermal and viscous processes on the surfaces of the different elements. The thermal energy being stored in the stack is proportional to the temperature difference across the stack. "Onset Delta T" refers to the temperature gradient necessary to reach spontaneous oscillations. The positioning of the stack and associated heat exchangers inside the resonator may be optimized so that spontaneous oscillations, at the resonance of the system, occurs at a minimum Delta T.

The resonator is a half-wavelength plane wave resonator. The fluid oscillates longitudinally at frequency f_0 determined by the resonator's length and gas properties. The resonant frequency of the system can be approximated by

$$f_0 = \frac{1}{\left(2\left[\frac{L_c}{c_c} + \frac{L_h}{c_h}\right]\right)}, \qquad 3.1$$

where L_c is the length of the ambient resonator, c_c is the sound speed in the ambient resonator, L_h is the length of the hot resonator, and c_h is the sound speed in the hot resonator.

3.2 Optimization of a Helium filled prime mover

The design criteria for the helium filled prime mover was to minimize the temperature difference required to reach onset. This is represented by onset Delta T. The heat flow (Equation 2.33) and work flow (Equation 2.34) of the short stack approximation presented in Chapter 2 were numerically evaluated, for chosen heat exchanger configuration; stack lengths and pore dimensions. The optimization consisted of adjusting the lengths of the hot and ambient resonator lengths, thus the position of the stack and heat exchangers, to obtain a minimum onset Delta T while holding the pore geometry constant.

The hot end of the resonator consists of a cylindrical brass tube with an inner diameter of 8.54 cm and 23.07 cm in length. The tube is flanged at one end with a 1.27 cm thick brass ring. A 1.90 cm brass plug is welded into the other end to produce a rigid end termination.

The ambient end is an aluminum tube 129 cm in length with an inner diameter of 8.52 cm. The tube is flanged on both ends with a 1.27 cm thick aluminum ring. One end is capped with a 1.27 cm thick aluminum disk which has a 0.635 cm microphone port.

The heat exchangers, shown in Figure 3.2, are 0.05 cm thick parallel plate copper fins spaced 0.102 cm apart encased in a 15.24 cm square copper block that is 1.90 cm thick. The hot heat exchanger has a 0.95 cm hole drilled on each side perpendicular to the fins. Cartridge heaters are inserted to supply heat. The cold heat exchanger has a an additional 0.95 cm hole drilled parallel to the fins to allow cooling fluid to be circulated by the ends of the fins. The flow rate can be adjusted to maintain the ambient heat exchanger at constant temperature.

The stack, as shown in Figure 3.3a and 3.3b, is a 5.08 cm long ceramic pore material containing 200 pores per square inch. The individual pores have square boundaries of 1.54 mm and extend the entire length of the stack. The stack has a diameter of 8.4 cm and is encased in a stainless steel cylinder flanged at both ends.

The theoretically predicted onset temperature difference and resonant frequency for Helium at atmospheric pressure in the optimized prime mover described above were 174 °C and 314.5 Hz respectively. The experimentally measured values are 178 °C and 314 Hz.

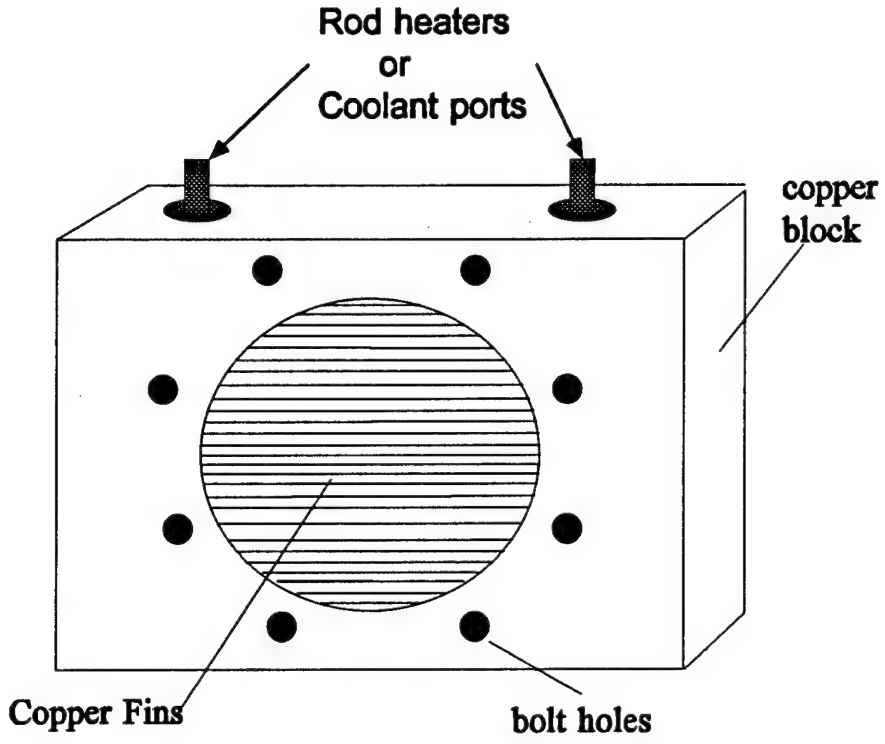

Figure 3.2. Parallel plate heat exchanger. The hot heat exchanger uses rod heaters inserted in the copper block to facilitate heat transfer to the system. The cold heat exchangers have coolant fluid circulating in the copper block to remove heat from the system.

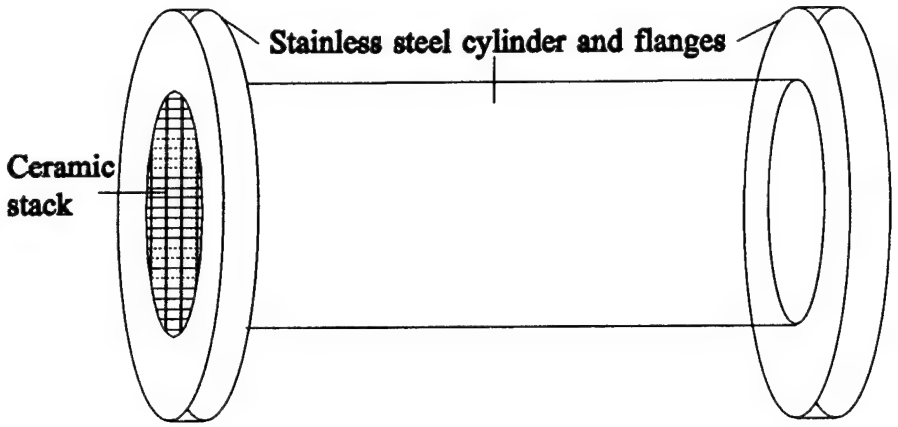

Figure 3.3a. The thermoacoustic stack and stainless steel encasement.

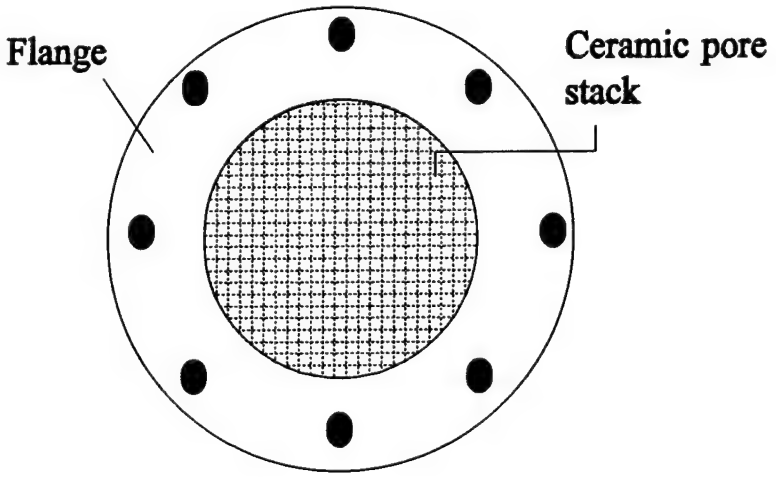

Figure 3.3b. Top view of the ceramic pore stack

3.3 Data Acquisition and Error Analysis

A. Temperature

The temperature gradient of the prime mover was established by heating the hot heat exchanger with two Omega Cir-2033/240 cartridge heaters. These heaters are positioned in the hot heat exchanger perpendicular to the ends of the fins. The electrical power was supplied by a Stanco variable ac transformer. The ambient heat exchanger was cooled by circulating water from a lab sink past the end of each fin. The temperature of water varied between 290-293K and the flow rate was adjusted to maintain the ambient temperature as close to 293K as possible.

The temperature of the heat exchangers were measured with a beaded type K thermocouple. The Omega 5TC-GG-K-10 thermocouple has a diameter of 0.025 cm and glass braid insulation. The response time is less than 0.5 sec and produces 40μV/°C with a standard error of 0.75%. The thermocouples are embedded in the heat exchangers midway between the heaters or coolant ports. The bead is pressed against the heat exchanger metal approximately 0.05 cm from the nearest fin. High temperature silicone was used to immobilize and isolate the thermocouple. Thermocouples (3) were also placed inside the resonator to be used at various positions. This was accomplished through a 0.35 cm hole in the ambient resonator wall which was sealed with epoxy.

In order to measure the temperature difference between various positions on the fins and the embedded thermocouples, other thermocouples were soldered to individual fins at different locations. The maximum difference in temperature was measured on the center of the middle fin. This difference was at most 3K. Slowly increasing the heat input and allowing the temperature to reach steady state before the addition of more heat decreases the difference to approximately ±0.5K.

The thermocouple voltage was monitored by a Omega HH82 digital thermometer calibrated in accordance with ANSI/mc 96.1. The HH82 digital thermometer has dual inputs, resolution of 0.1°C, accuracy ±0.2°C and displays the temperature or temperature difference of two thermocouples. The relative error between the two thermocouples is less that ±0.1°C at ambient temperature and increases to ±0.2°C at 300°C. The reproductivity of the temperature measurements of the heat exchangers is within ±1°C at high temperatures and decreases to ±0.1°C at ambient temperatures.

B. Mean and Acoustic Pressure

The modular design adopted in the construction allows for the modification of the individual elements or the system as a whole. Each element is designed to bolt to any other element or additional elements may be added. High temperature silicone sealant is used in all the junctions to ensure proper seals.

The prime mover was connected to a gas handling system through a 0.95 cm port located at the acoustic pressure node. An Omega model DPG-500 pressure gauge, Welch vacuum pump and various Matheson gas regulators were attached via a Richie Yellow Jacket manifold. This location was chosen because it is the least intrusive and mean pressure could be monitored and adjusted if necessary with very little effect on the onset Delta T.

The acoustic pressure was monitored by a Panasonic model 34E2S electret microphone located at the end of the ambient temperature resonator section. The microphone output was ac coupled to a Hewlett Packard model 35665A dynamic signal analyzer. This microphone was sufficient for detection of onset of oscillation but saturates at relatively low amplitudes. A comparison of the electret microphone and an Endevco model 8510B-5 piezoesistive microphone was made and the reproductibility of the measured onset temperature difference was within ±0.2K. There was no noticeable difference in the detection of onset of acoustic oscillations.

3.4 Stability Analysis of a Helium filled Prime mover

A. Stability Curve

The boundary between damped and excited oscillations is defined as the stability curve. The stability curve represents the temperature difference necessary to just match the total losses in the prime mover, that is, the time averaged energy

exchanged between the working fluid and the stack is zero. The temperature difference required for a helium filled prime mover to reach onset can be varied with the viscous and thermal penetration depths.

The thermal and viscous penetration depths can be written for an ideal gas as

$$\delta_\kappa = \left(\frac{2k}{\rho_0 \omega c_p}\right)^{1/2} \propto (\omega P_0)^{-1/2}, \qquad 3.2$$

and

$$\delta_\nu = \left(\frac{2\eta}{\rho_0 \omega}\right)^{1/2} \propto (\omega P_0)^{-1/2} \qquad 3.3$$

where k is the thermal conductivity, ρ_0 is the ambient density, ω is the angular frequency, P_0 is the ambient pressure, c_p is the specific heat at constant volume and η is the coefficient of viscosity. The thermal and viscous penetration depths can be adjusted by changing the ambient pressure for a fixed length resonator.

Figure 3.4 shows the experimental configuration used to measure the stability curve for a helium filled thermoacoustic prime mover. The data acquisition and error analysis of sections 3a and 3b was applied here with the following exceptions. The

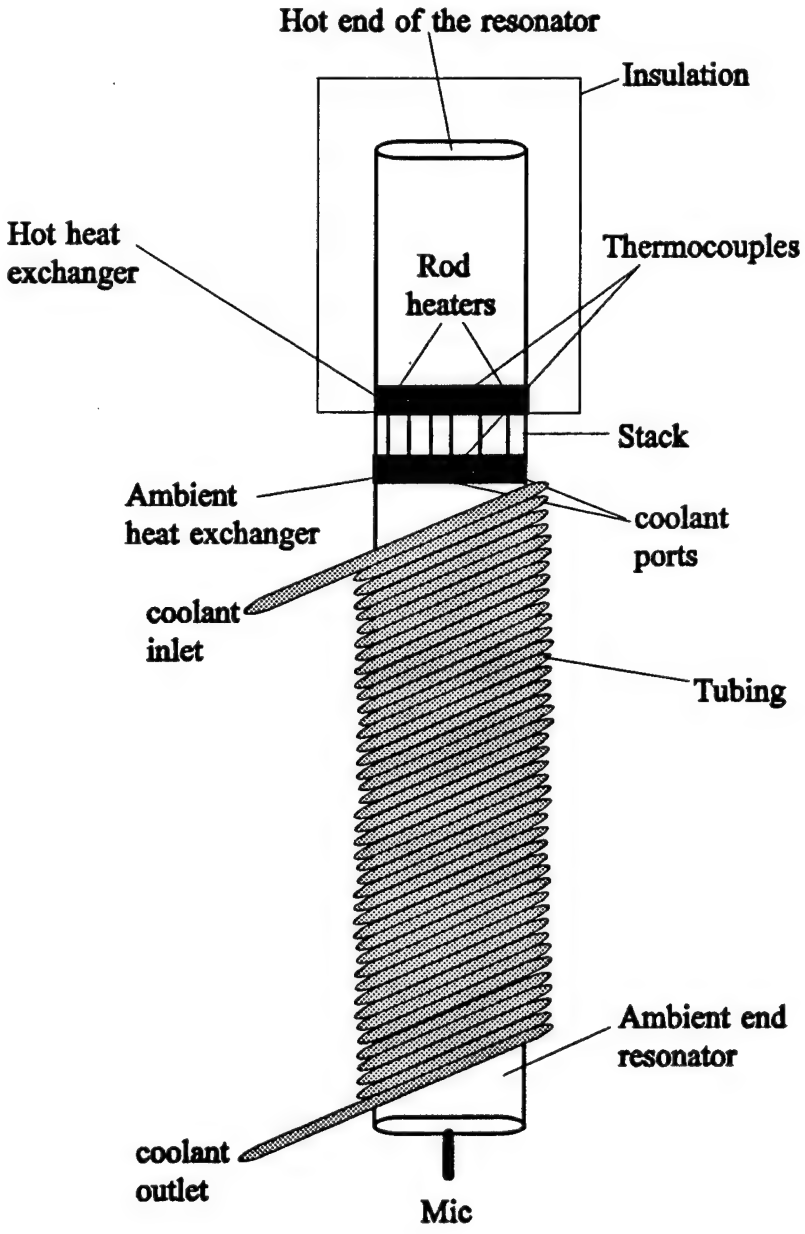

Figure 3.4. The experimental configuration for the stability curve analysis.

ambient heat exchanger and ambient end of the resonator were maintained at 20°C with water circulating through a Cole-Parmer G-12101-10 constant temperature bath. The resonator was evacuated then filled with helium to the desired mean pressure. The temperature at the hot end was increased slowly until sustained self-oscillation was detected via a microphone located in the end plate of the ambient resonator. The temperature difference between the heat exchangers was then recorded along with the frequency of self oscillation and ambient pressure. The resonator was then cooled down, the ambient pressure changed, and the process repeated. Measurements were performed for ambient pressures of 143, 170, 184, 198, 212, 239, and 308 kPa. The experimental values shown in Figure 3.5a and 3.5b are in good agreement with the theoretical curve, but is shifted upwards by approximately four degrees.

B. Short Stack approximation for the Stability Curve[19,26]

Work flow as computed from the short stack approximation in Chapter 2 can be used to derive the optimal position of a stack in the standing wave and the corresponding minimal onset temperature difference. The gas properties are assumed to be constant and are evaluated at the temperature of the hot end of the stack.

A nondimensional critical temperature gradient can be defined as

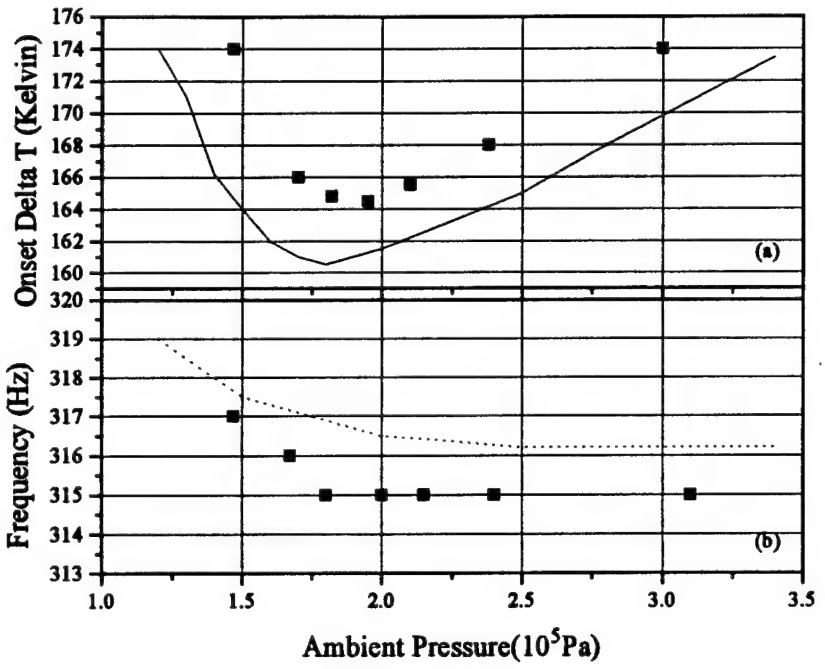

Figure 3.5(a) Shows the onset Delta T for the helium prime mover and (b) the associated frequency. The solid and dotted lines are the theoretical predicted values.

$$\tau = \frac{\beta(T_H - T_c)}{2k_0 d}, \qquad 3.4$$

where T_H is the temperature of the hot heat exchanger and T_C is the temperature of the ambient heat exchanger, β is the coefficient of thermal expansion, k_0 is the adiabatic propagation constant for sound speed c and operation frequency f_0 and d is the stack length. τ is a nondimensional measure of the temperature gradient necessary for thermoacoustic power production to just match the sum of the power dissipation elsewhere in the prime mover.

The work flow in the short stack approximation is given by Equation 2.54. The first and second terms, always negative, are general expressions for dissipation of potential and kinetic energy per unit time due to thermal and viscous diffusion processes. This is applicable to arbitrary pore diameters and shapes. The third term, which is positive when the hot end faces a pressure antinode, is the acoustic power produced by the temperature gradient. When the third term is larger than the sum of the first two the stack produces net acoustic power. The amount of loss by thermal or viscous dissipation and gain by the thermoacoustic mechanism is weighted by the position of the stack in the standing wave.

Denote the total rate of work loss in the heat exchangers and the rest of the resonator as $\overline{\dot{W}}_{ext}$. The oscillations can occur when

$$\overline{\dot{W}}_s + \overline{\dot{W}}_{ext} \geq 0. \qquad 3.5$$

It is useful to nondimensionalize $\overline{\dot{W}}_{ext}$ by defining

$$\overline{\dot{w}}_{ext} \equiv \frac{2\rho_0 c^2 \overline{\dot{W}}_{ext}}{p_1^2(0) V_G \omega} = \frac{2\gamma p \overline{\dot{W}}_{ext}}{p_1^2(0) V_G \omega}. \qquad 3.6$$

The second form assumes an ideal gas and is independent of the ambient temperature.

The onset temperature gradient for the thermoacoustic engine can generally be evaluated using Equations (3.4) , (2.34), and (3.5). Evaluation of the onset condition in Equation (3.5) yields

$$\tau = \frac{\Omega(1-N_{pr})}{\mathrm{Im}\left(F^*(\lambda_T)/F^*(\lambda)\right)}\left(\frac{\overline{\dot{w}}_{ext}}{\sin(2\phi}+\frac{(\gamma-1)\mathrm{Im}\,F^*(\lambda_T)}{2\tan\phi}+\frac{\tan\phi\,\mathrm{Im}\,F^*(\lambda_T)}{2|\Omega F(\lambda)|^2}\right). \qquad 3.7$$

The phase angle and external work flow were computed to be $\varphi = k_0 z = 27.2$ deg. and $\overline{\dot{w}}_{ext} = 0.52$ for λ_T=2.4. This is the value of λ_T corresponding to the minimum experimental onset temperature difference. The short stack approximation (Eq. 3.7)

for τ is shown along with experimental data in Figure 3.6 verifying the utility of the approximation. It should be noted that both φ and $\overline{\dot{w}}_{ext}$ depend on λ_T.

The stack placement in the resonance tube for minimum onset temperature gradient can be determined by the additional condition that τ be minimized with respect to phase angle. The phase angle for minimum onset temperature gradient, with the assumption that $\overline{\dot{w}}_{ext}$ varies negligibly with z, is determined from

$$\tan(\varphi_{min}) = \sqrt{\frac{(\gamma-1)\,\mathrm{Im}\,F^*(\lambda_T) + \overline{\dot{w}}_{ext}}{\mathrm{Im}\left[F^*(\lambda)\Big/|\Omega F(\lambda)|^2\right] + \overline{\dot{w}}_{ext}}}. \qquad 3.8$$

The corresponding minimal nondimensional temperature gradient is given by

$$\tau_{min} = \left(\frac{\Omega(1-N_{pr})}{\mathrm{Im}\left(F^*(\lambda_T)\Big/F^*(\lambda)\right)}\right) \Bigg/ \sqrt{\left((\gamma-1)\,\mathrm{Im}\,F^*(\lambda_T) + \overline{\dot{w}}_{ext}\right)\left(\mathrm{Im}\left[\frac{\mathrm{Im}\,F^*(\lambda_T)}{|\Omega F(\lambda)|^2}\right] + \overline{\dot{w}}_{ext}\right)}. \qquad 3.9$$

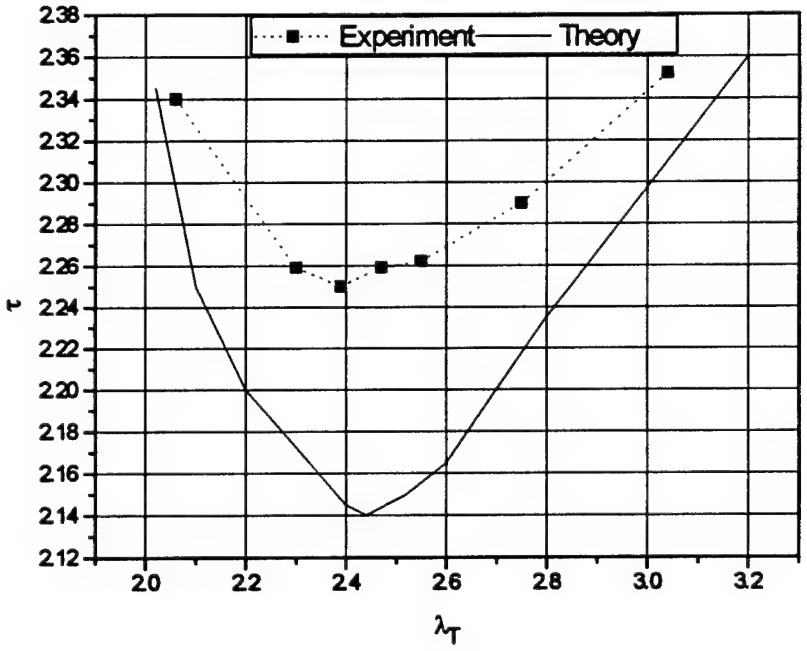

Figure 3.6. The nondimensional temperature gradient vs. λ_T.

Equations 3.8 and 3.9 illustrate the role played by the thermoviscous function $F(\lambda)$ in thermoacoustics. When $\overline{\dot{w}}_{ext} > 1$, $\varphi_{min} = \pi/4$ and

$$\tau_{min} = (1-N_{pr})\ \overline{\dot{w}}_{ext}\ \Omega\ /\{\mathrm{Im}(F^*(\lambda_T)/F^*(\lambda)). \qquad 3.10$$

The losses in the stack are much less than the acoustic load elsewhere so the stack is to be positioned for maximum effectiveness of the thermoacoustic gain term, i.e., where the product of acoustic pressure and particle velocity are a maximum; a distance 1/8 of the acoustic wavelength from the hot end. For an inviscid gas, $F(\lambda)=1$, and for $\overline{\dot{w}}_{ext} = 0$, there is no acoustic load and $\varphi_{min} = \pi/2$. The stack is at the pressure node a distance of 1/4 of an acoustic wavelength from the hot end. In a realistic engine the viscous losses in the stack and heat exchangers dominate, and the optimum stack position is displaced from $\pi/4$ toward the end of the tube.

Chapter 4

Working Fluid Interactions

4.1 Introduction

The primary elements in a thermoacoustic device are the resonator and the working fluid. The stack is used to localize the interactions with the working fluid and to increase the efficiency of the interactions. Work flow in the short stack approximation to the first order in $k_0 d$ can be written as[19]

$$\begin{aligned}\overline{\dot{W}_2}(z) = &-\omega \frac{V_G P_1^2(z)}{2\rho_0 c^2}(\gamma - 1)\mathrm{Im}F^*(\lambda_T) \\ &-\omega \frac{\rho_0 V_G v_z^{s\,2}(z)}{2} \frac{\mathrm{Im}F^*(\lambda)}{|F(\lambda)|^2} \\ &+\frac{V_G}{2} P_1(z) v_z^s(z) \beta \frac{(T_H - T_C)}{d} \frac{\mathrm{Im}[F^*(\lambda_T)/F(\lambda)]}{1 - N_{pr}}.\end{aligned} \tag{4.1}$$

The first and second terms are dissipation of potential and kinetic energy due to the interaction of the working fluid with the surface area of the stack within the respective penetration depths. These terms are always negative. The last term is the acoustic power produced due to the temperature gradient across the stack and is always

positive. We can see that the gas properties play an important role in both dissipation and production of acoustic power.

The work produced in a thermoacoustic prime mover's stack is proportional to γ-1, where γ is the ratio of the specific heats, this term is commonly referred to as the work parameter of the fluid. Gamma can be approximated by $\gamma \cong 1+2/N$, where N is the number of degrees of freedom of the gas as shown in Figure 4.1. For a monatomic gas $N = 3$ and $\gamma \cong 5/3$, while for other gases $N > 3$ and $\gamma < 5/3$.

Another important property is the Prandtl number, $N_{pr} = \eta c_p/\lambda$, where η is the viscosity, c_p is the specific heat at constant pressure, and λ is the thermal conductivity. λ is used for the thermal conductivity to avoid confusion with Boltzmann's constant. Eucken[30] has shown that the thermal conductivity of a pure gas can be approximated as $\lambda \cong \eta c_v(9\gamma-5)/4$. The Prandtl number can then be written as $N_{pr} \cong 4\gamma/(9\gamma-5)$ and is shown in Figure 4.1. For monatomic gases $N_{pr} \cong 0.68$ and for a more complicated gas approaches one, as γ approaches one.

The ideal working fluid in thermoacoustics would have a large γ and small N_{pr}. This would maximize the thermodynamic process on which the prime mover is based, thermal conduction, while the viscous losses would be reduced. Many pure gases meet the condition of high γ, but not small N_{pr}. Binary gas mixtures are one way to achieve both conditions.

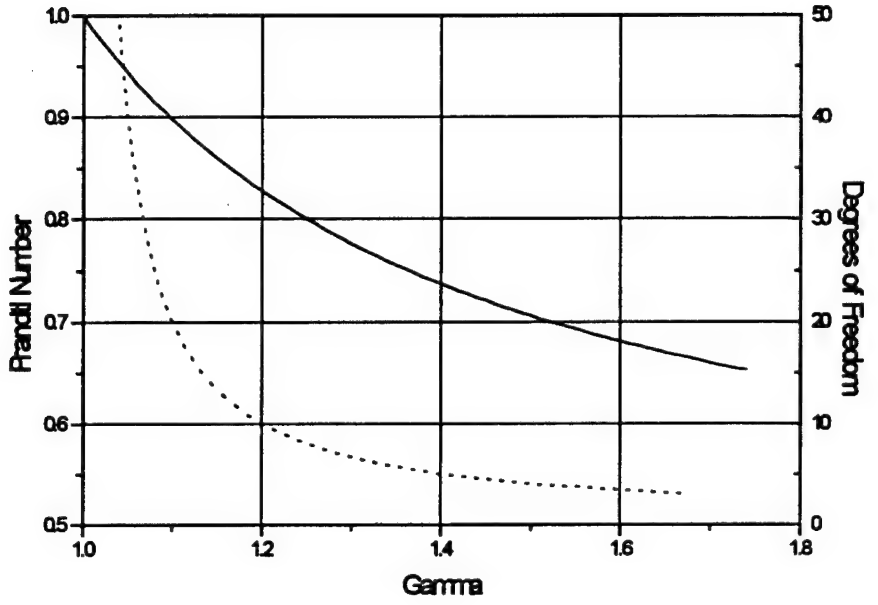

Figure 4.1. The Prandtl number vs gamma for pure gases is indicated by – and ··· shows the dependence of gamma on the degrees of freedom.

4.2. Binary Gas Mixtures

A. Simplified Theory

The formulation of the simplified theory for the properties of binary gas mixtures is presented in Appendix D section 1a. The Prandtl number for binary gas mixtures can be approximated by using Equation D.11, Equation D.12, and the specific heat at constant pressure. The specific heat at constant pressure for a binary monatomic gas mixture is given by Equation D.2 and can be expressed as

$$\left[c_p\right]_{mix} = \frac{R\left(1+\frac{N}{2}\right)}{M_1 x_1 + M_2 x_2} = \frac{R\left(1+\frac{N}{2}\right)}{M_1\left(x_1 + \frac{M_2}{M_1}(1-x_1)\right)} \qquad 4.2$$

where N is the number of degrees of freedom, R is the gas constant, M_1 and M_2 are the molecular weights of species one and two respectively, and x_1 and x_2 are the mole fraction of the species.

The Prandtl number using Equations 4.2, D.14, and D.16 can be written as

$$\left[N_{pr}\right]_{mix} = \frac{2}{5}\left\{\frac{\dfrac{R\left(1+\dfrac{N}{2}\right)}{M_1\left(x_1+\dfrac{M_2}{M_1}(1-x_1)\right)}\sqrt{M_1}\left(x_1+\sqrt{\dfrac{M_2}{M_1}}(1-x_1)\right)}{\dfrac{c_v}{\sqrt{M_1}}\left(\dfrac{1}{x_1}+\dfrac{1}{x_2\sqrt{\dfrac{M_1}{M_2}}}\right)}\right\}. \qquad 4.3$$

This suggests that the Prandtl number is dependent on the ratio of the molecular weights of the gases present in the mixture. Monatomic gases have 3 degrees of freedom, i.e. N=3, and $c_v = c_p$-R. The Prandtl number for binary mixtures of helium-neon, helium-argon, and helium-krypton are shown in Figure 4.2. These monatomic gases have approximately the same Prandtl number and gamma in their pure states. Binary mixtures produce a decrease in the Prandtl number with a minimum occurring when the lighter gas is approximately 60% of the molecular volume. The mixture of helium-neon has a mass ratio of 5 and provides the smallest change to the Prandtl number. The mixture of helium-krypton has a mass ratio of 16 and has the largest affect on the Prandtl number.

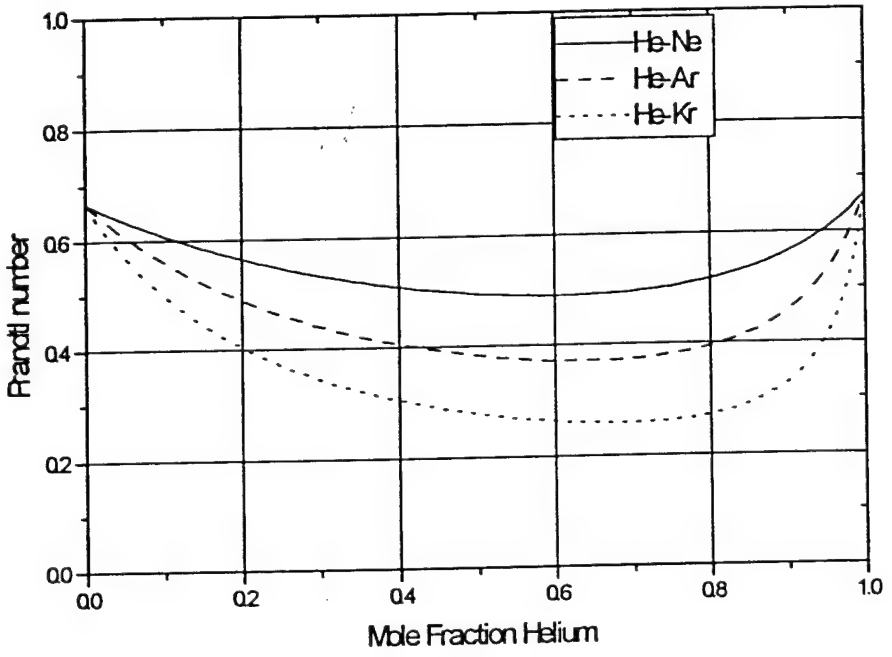

Figure 4.2. Binary gas mixtures using the simplified model.

B. Kinetic Transport Theory

A formulation of the transport properties of viscosity and thermal conductivity for a gas mixture composed of ν components is presented in Appendix B. Binary gas mixture are a special case of this theory, with $\nu = 2$ and is presented in Appendix D. A FORTRAN routine to evaluate the Prandtl number and gamma of binary mixtures based on this formulation is presented in appendix E. The numerous parameters necessary for the Prandtl number calculation for the different gases considered for thermoacoustic devices are presented in Table 4.1.

Theoretical predictions of the Prandtl number for helium mixed with argon, krypton, and xenon are shown in Figure 4.3 for 300K. A mixture containing 60 percent helium and 40 percent of any of the other gases produces a minimum in the Prandtl number. The maximum decrease of 67 percent occurs for a helium-xenon mixture. The simplified model (section 4.2a Figure 4.2) produces the same general trend in the Prandtl number as the more complicated transport theory calculations with only about 10% difference. This is shown in Figures 4.2 and 4.3. Figure 4.4 shows little change in the ratio of specific heats (gamma) with the concentration since all gases are monatomic.

Comparing the Prandtl numbers for a helium-argon mixture from figure 4.3 and

		Ref. 28	Ref. 28	Ref. 32	Ref. 32	Ref. 32	Ref. 32	Ref. 32
Gas	Mass (gram)	ε/κ (K)	σ (Å)	c_p cal/ m K	c_v cal/ m K	γ	κ cal/ sec cm K	μ 10^{-4} poise
Helium	4	10.22	2.566	4.968	3.0478	1.63	.00035	1.96
Neon	20.18	35.7	2.789	4.968	3.026	1.642	.000117	3.173
Argon	39.94	124	3.418	5.000	2.996	1.668	.0000415	2.2638
Kr	83.80	190	3.61	5.328	3.1968	1.67	.0000228	2.53
Xenon	131.3	229	4.055	5.016	3.02	1.66	.0000130	2.31
Air	29	84	3.698	6.96	4.97	1.4	.000062	1.846
SF_6	146	200.9	5.51	23.26	22.36	1.04	.000033	1.53
O_2	32	113	3.433	7.019	5.028	1.396	.0000639	2.063

Table 4.1 Gas Properties

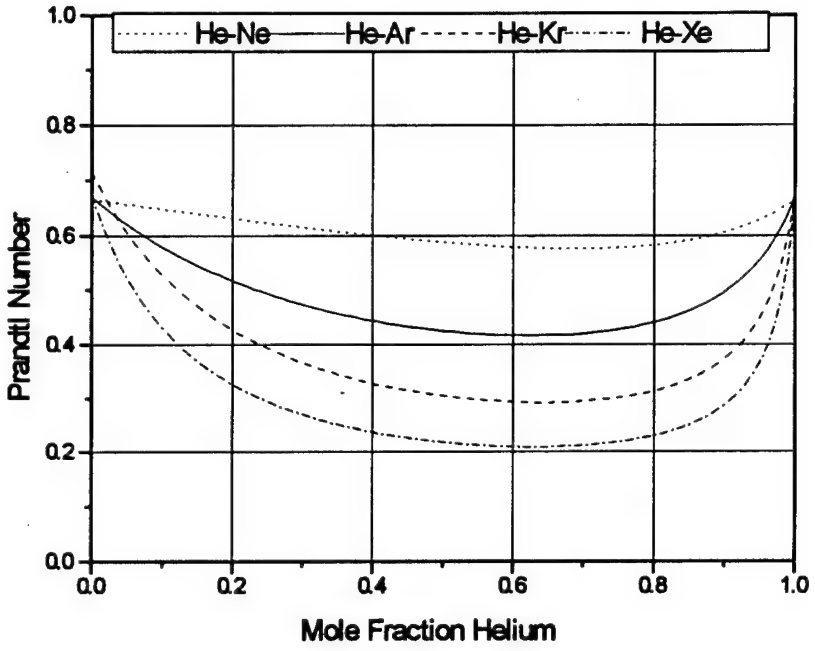

Figure 4.3. The Prandtl number for binary mixtures of helium with neon, argon, krypton, and xenon at 1 atm. and temperature of 300K.

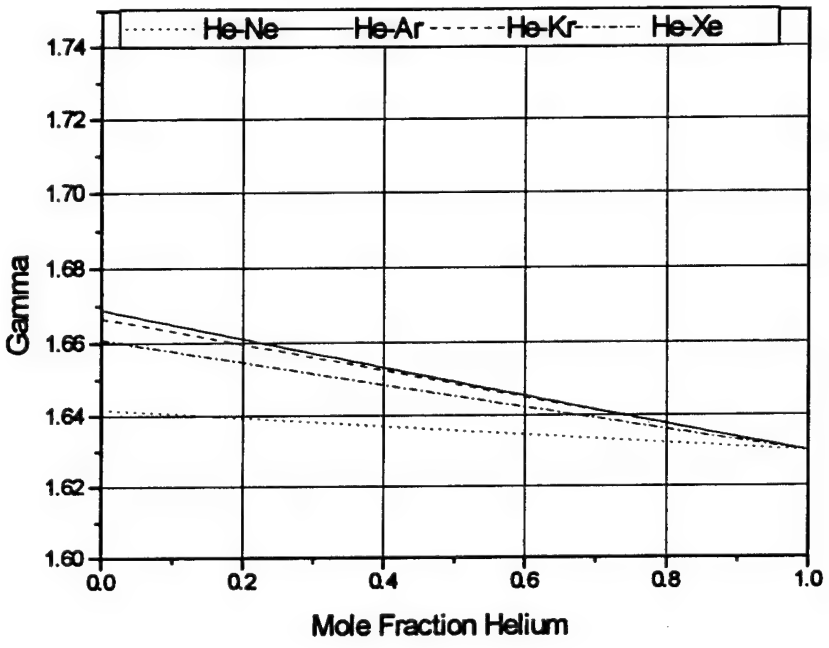

Figure 4.4. The ratio of the specific heats for binary mixtures of helium and the other noble gases at 300K and 1 atm.

the air-helium mixture from Figure 4.5 show that the two closely resemble each other. The minimum in the air-helium mixture curve is shifted by 2 percent, with a minimum Prandtl number of 0.44 compared to 0.42 for helium-argon. Comparing values for helium-neon mixtures and air-krypton mixtures, the variation in the Prandtl number is negligible. This observation suggests that the Prandtl number for binary gas mixtures is mass ratio dependent as was suggested in Figure 4.2. The larger species' mass should be a minimum of five times larger than the smaller species in the mixture for any useable affect on the Prandtl number.

Figure 4.5 shows Prandtl number predictions for mixture of air with helium, argon, neon, and krypton at 300K. Air is assumed to be diatomic. The Prandtl number varies very little for air mixed with argon, neon, and krypton, but varies by 34 percent for a 42 percent air-58% helium mixture. Gamma decreases linearly from the noble gas value to the 1.4 value for air.

The larger the molecular mass of the heavier species the smaller the amount needed to decrease the Prandtl number. Figure 4.6 shows the Prandtl number and gamma for a gas mixture of helium and sulfur hexafluride (SF_6), which is 36.5 times heavier than helium. A mixture containing 10 percent SF_6 decreases the Prandtl number by 45 percent with gamma only decreasing by 16 percent. The minimum Prandtl number occurs for a mixture of 35 percent SF_6 and 65 percent helium

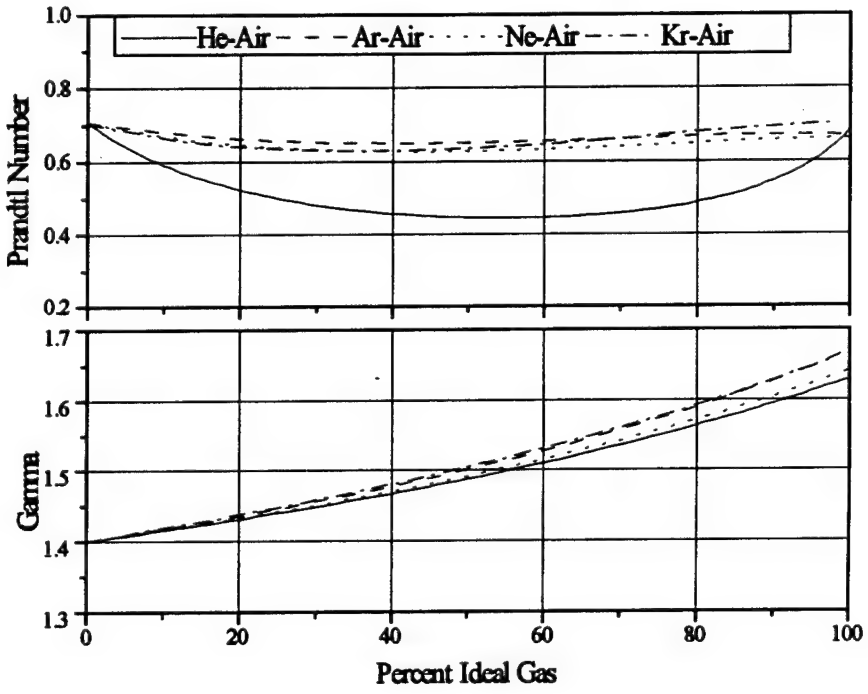

Figure 4.5 Prandtl Number and ratio of specific heats for air mixed with different ideal gases.

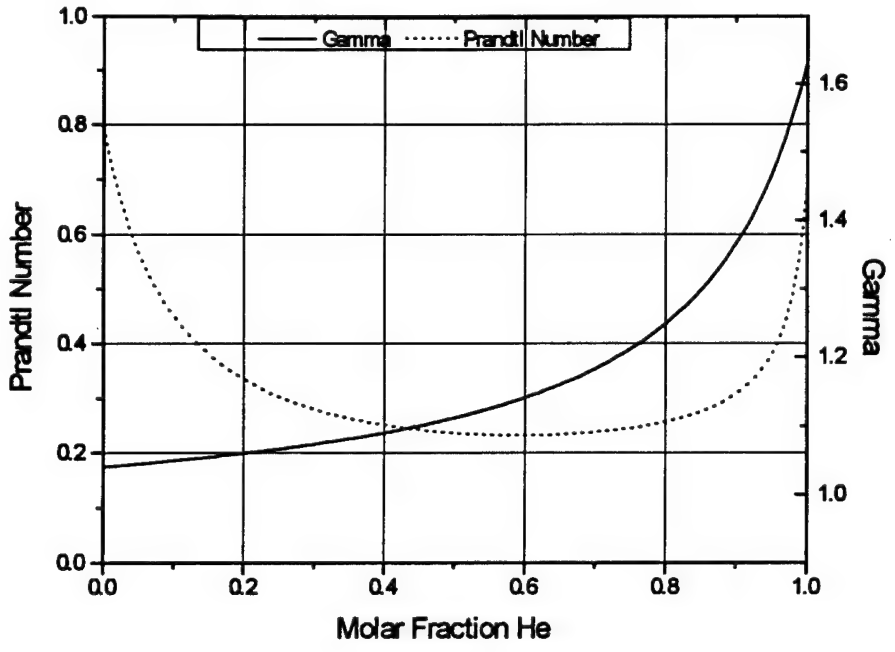

Figure 4.6. The Prandtl number and ratio of specific heats for mixtures of helium and SF_6 at 300 K and 1 atm.

4.3 Experimental Results for Binary Gas Mixtures

Binary gas mixtures of helium-argon and helium-SF_6 were shown in the previous section to have properties necessary as working fluids in thermoacoustic devices. The prime mover shown in Figure 3.4 was used to experimentally observe changes in onset temperature when the working fluid was a binary mixture. The heat exchangers used were 0.32 cm in length with a fin spacing of 0.05 cm and the hot and cold resonator sections were increased by 1.59 cm to maintain the optimum position of the stack. The data acquisition and error analysis discussed in Chapter 3 sections 3a and 3b are applicable here.

The prime mover was evacuated and filled with helium. The second gas was added until the desired percentage was reached using the law of partial pressures. The gases were allowed to mix then excess pressure was released. The temperature gradient was then increased until onset of self oscillation was detected. The temperature, frequency and percentage of the gases were recorded. The prime mover was cooled to room temperature and the process repeated.

Pressurizing the prime mover above the desired pressure during mixing was done for two reasons. First, it allowed for greater accuracy in the calculation of the percentages of each gas. Second, if the frequency shifted while venting the excess

pressure, the gases were not thoroughly mixed and the procedure was repeated.

Figure 4.7 shows the theoretical and experimental onset temperature and frequency for various mixtures of helium and SF_6. The minimum value for the Prandtl number is shown in Figure 4.6 to occur at 35 percent SF_6 which corresponds to the minimum observed onset temperature difference of 77K. The agreement between theory and experiment is quite good for low percentages of SF_6, but at high percentages, the two values deviate considerably. This may be due to the temperature dependence of the properties in the mixture, which in the calculations was approximated by the temperature dependence of the properties of helium.

Figure 4.8 shows the theoretical and experimental onset temperature and frequency for various mixtures of helium and argon. The minimum value for the Prandtl number occurs in the range of 35-45 percent argon, which corresponds to a minimum onset temperature difference of 125K. Theoretical calculations and experimental data are in good agreement with the largest difference of 5 percent for Delta T and 3 percent for frequency.

Figures 4.7 and 4.8 show the onset temperature differences needed for self oscillation for pure helium (Delta T=170K), argon (Delta T=130K), and SF_6 (Delta T=108K). With a Prandtl number equal to 0.8 and gamma of 1.04 pure SF_6 has the lowest onset temperature. Argon has approximately the same Prandtl number and

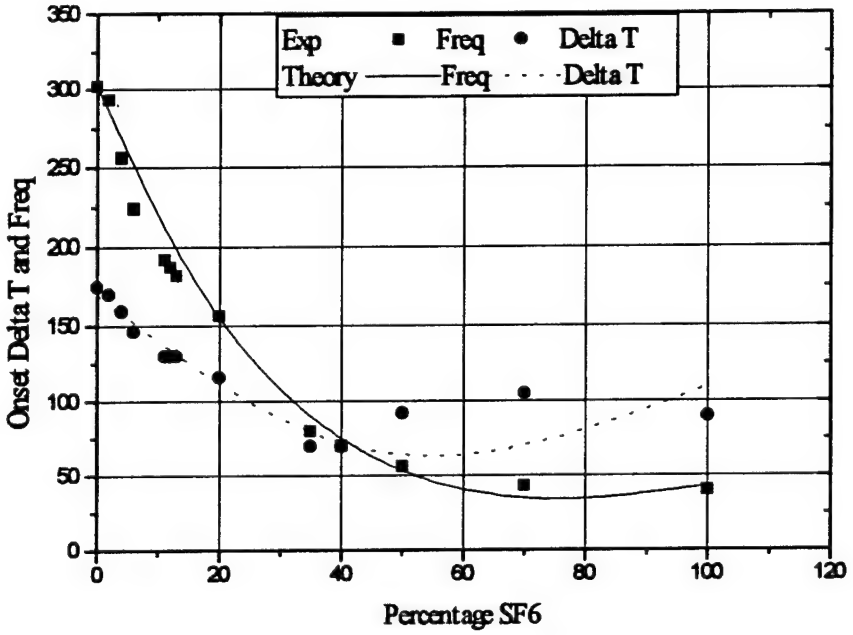

Figure 4.7. The onset temperature difference in Kelvin and frequency in hertz vs the percentage of SF_6 mixed with helium at 1 atm. The temperatures were measured in the heat exchangers.

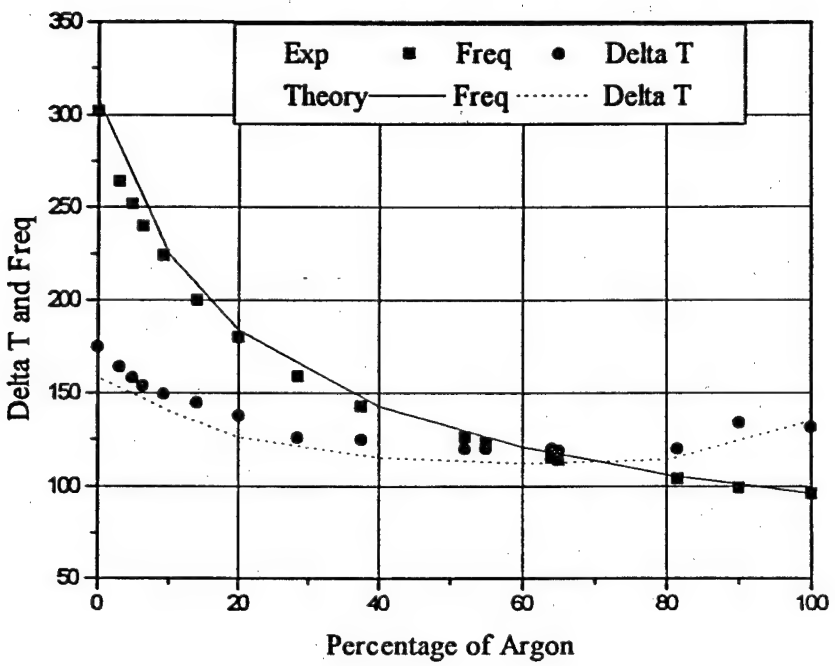

Figure 4.8. The onset temperature difference in Kelvin and frequency in hertz vs the percentage of argon mixed with helium at 1 atm. The temperatures were measured in the heat exchangers.

gamma as helium but the onset temperature is 40 degrees lower. These results appear to contradict conventional wisdom which suggest that a large gamma gives best results.

4.4 Physically Changing the Interaction Length

The interaction of the working fluid with the stack produces thermal and viscous losses, as shown in the first two terms of Equation 4.1. These losses are proportional to the length d of the stack. The production of work, represented by term three of Equation 4.1, is inversely proportional to the length. By physically changing the length of the stack the interaction between the stack and the working fluid will be explored.

The prime mover shown in Figure 3.4 was used to investigate the effects of changing the length of the stack. The stack position was maintained at the optimal location by varying the hot end and ambient end resonator lengths by half the length added or removed from the stack, keeping the system frequency constant.

The experimental results are shown in Figure 4.9 along with the theoretical prediction for helium and air as the working fluids. The physical constraints of the prime mover limited the smallest experimental stack to 2.54 cm, which resulted in the lowest onset temperature for both helium and air. The experimental and theoretical result agree. Theoretically there should be a minimal stack size where heat conduction

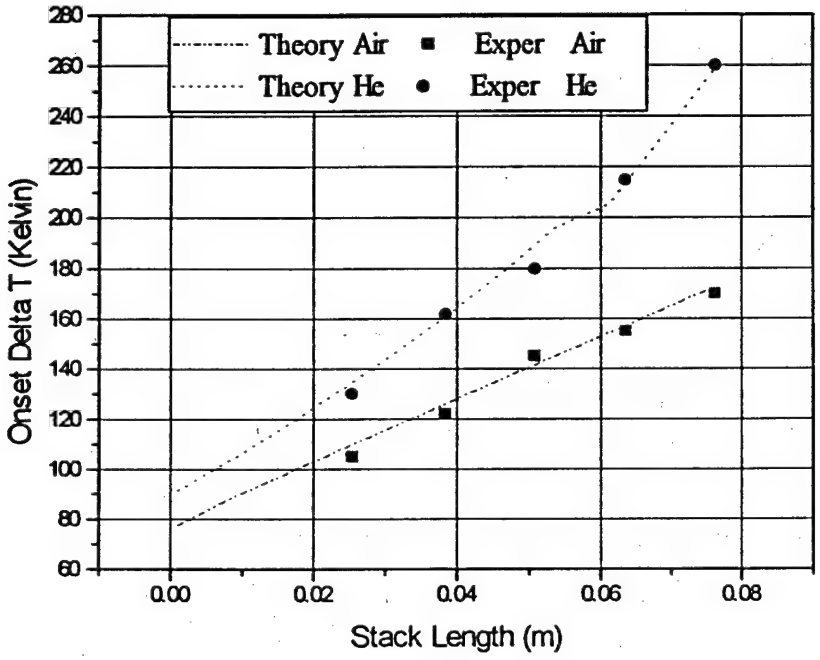

Figure 4.9. The temperature necessary for onset of self oscillations vs the length of the stack interacting with the working fluid. Experimental stack lengths of 0.0254, 0.0381, 0.0508, 0.0635, and 0.0762 meters are shown.

through the working fluid and stack holder prohibits maintenance of a temperature gradient. For a 5 cm length stack the cold heat exchanger must carry away approximately 16 watts of conducted heat to maintain its temperature. For a stack of 0.5 cm this conducted heat is 160 watts and quickly becomes unmanageable.

Chapter 5

Heat Transfer to the Working Fluid

5.1 Introduction

Onset of self oscillation occurs when the temperature gradient is large enough across the stack to produce a balance in the energy of the system. Heat exchangers are used in thermoacoustics to supply and extract heat at the ends of the stack to maintain the temperature gradient. Their presence, in a prime mover, is purely dissipative in both viscous and thermal processes. The commonly used heat exchanger in thermoacoustics consists of parallel copper plates (fins) heated or cooled externally as shown in Figure 5.1

The fundamental mechanisms for heat transfer are conduction, convection, and radiation. The parallel plate heat exchanger is primarily a heat conducting device prior to onset. Following onset, heat conduction occurs in the thermal boundary layer surrounding the individual fins of the heat exchanger and is shown in Figure 5.2. The working fluid within this boundary is near the same temperature as the fins of the heat exchanger when the plate separation is less than $2\delta_\kappa$.

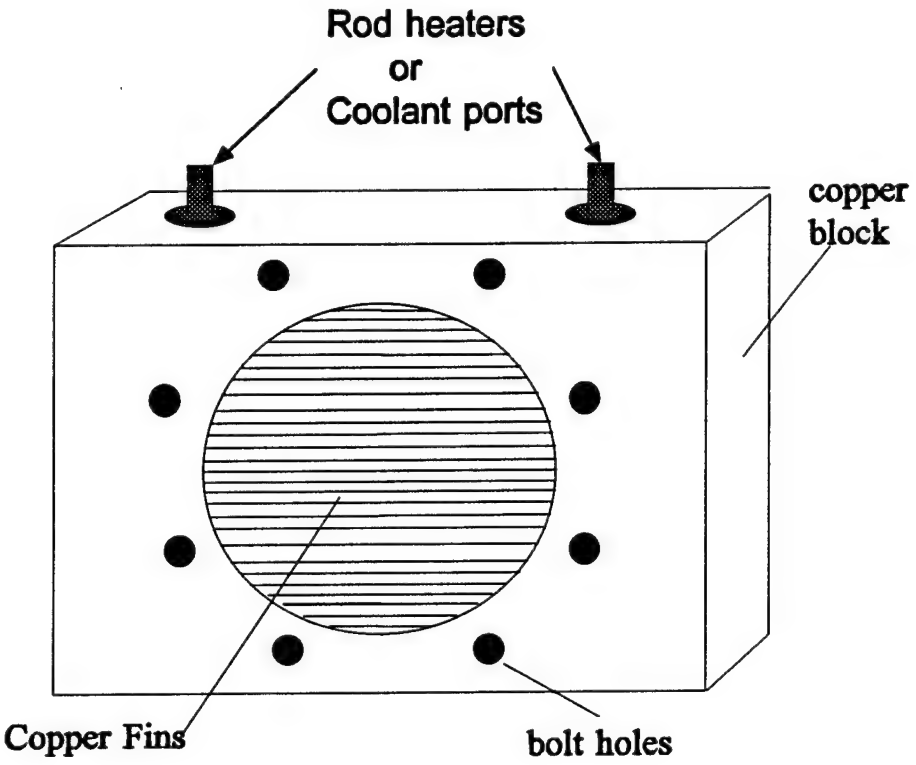

Figure 5.1. Parallel plate heat exchanger. Rod heaters are used to conduct heat into the working fluid and coolant fluid is used to extract heat. The copper block is 15.24 cm square and 1.905 cm thick. The distance between the fin is denoted by R that is 0.1016 cm and the plate thickness is l = 0.0508 cm.

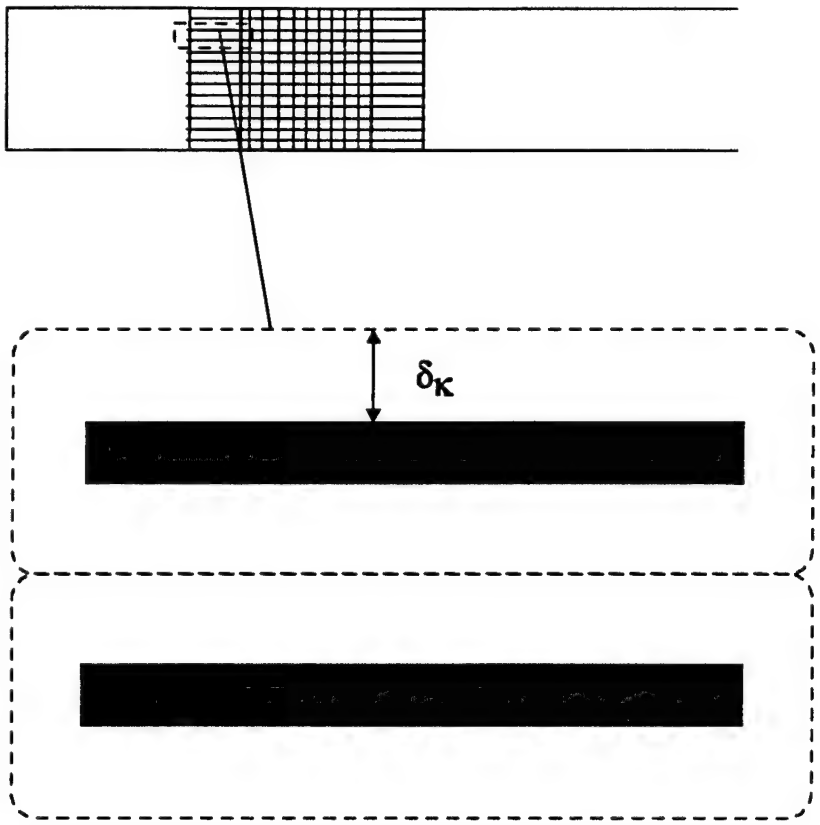

Figure 5.2. At onset the primary mode of heat transfer is conduction. The working fluid in thermal contact with the fins of the heat exchanger is within the thermal penetration depth δ_κ.

Fins are used to increase the amount of gas in thermal contact with the heat exchangers and are necessary to maintain the temperature difference across the stack required to reach onset of self oscillation. The role of the heat exchangers is studied in this section.

Referring to Figure 5.3, the temperature difference which results in self oscillation (onset Delta T) is represented by the y axis and the heat exchanger configuration is represent by the x axis. The prime mover described in Chapter 3, section 2 is represented by heat exchanger configuration 1. The fins were removed from the hot heat exchanger in configuration 2. In configuration 3, the fins are replaced in the hot heat exchanger and removed from the cold heat exchanger. Both heat exchangers are finless in configuration 4. The working fluids are air and helium.

Type K thermocouples were embedded in the hot and cold heat exchangers and the onset Delta T_h is the difference between these two temperatures. This temperature difference is proportional to the thermal energy supplied to the prime mover. The data acquisition and error analysis discussed in Chapter 3 sections 3a and 3b is the same here, with the exception that two type K thermal couples were also positioned at the ends of the stack to measure the temperature of the working fluid at the entrance to the stack. The temperature difference measured on the stack ends is represented by onset Delta T_s.

The data presented in Figure 5.3 for heat exchanger configuration 1 shows a difference of 10K between the onset Delta T_h and Delta T_s. This difference is present for both helium and air. The heat exchangers are in physical contact with the ends of the stack as shown in Figure 5.2. As a result the thermal penetration depth around the end of the fins extends into the stack. Therefore, the temperature of the working fluid in the heat exchanger should be approximately the same as the working fluid near the end of the stacks. Figure 5.4 shows the temperature distribution along the stack and across the stack/heat exchanger interface. Note the temperature jump at the stack-heat exchanger junction. The thermocouple could be in partial contact with a pore wall in the stack which might account for the temperature jumps at the interfaces.

Results for heat exchanger configurations 2, 3 and 4 presented in Figure 5.3 show that the removal of the fins from one or both of the heat exchangers affects the temperature difference required for onset. Prior to onset, the heat exchanger with no fins relies on thermal conduction of the working fluid to transport heat to and from the stack. Gas in the vicinity of the stack ends serves as a heat reservoir.

The temperature difference across the stack required to initiate self oscillation decreases with the removal of fins from either or both of the heat exchangers. The decrease in Delta T_s is a result of the reduced attenuation in the heat exchangers. The equal drop in Delta T_s for configuration 2 and 3 shows that the hot and cold heat exchangers have approximately the same losses. The heat exchangers used account

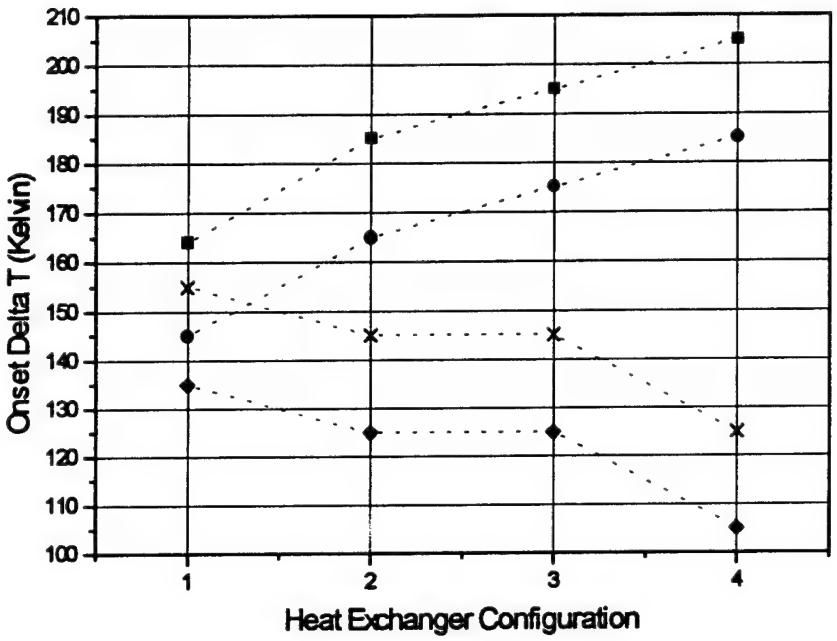

Figure 5.3. Onset Delta T vs. Heat exchanger configuration. The x axis represents the heat exchanger's configuration: 1 both heat exchangers are in the system, 2 no hot heat exchanger, 3 no ambient heat exchanger, and 4 represent no heat exchangers in the system. The measured temperature difference between the heat exchangers (Delta T_h) is represented by ■ for helium and ● for air. The measured temperature difference across the stack (Delta T_s) is represented by x for helium and ◆ for air.

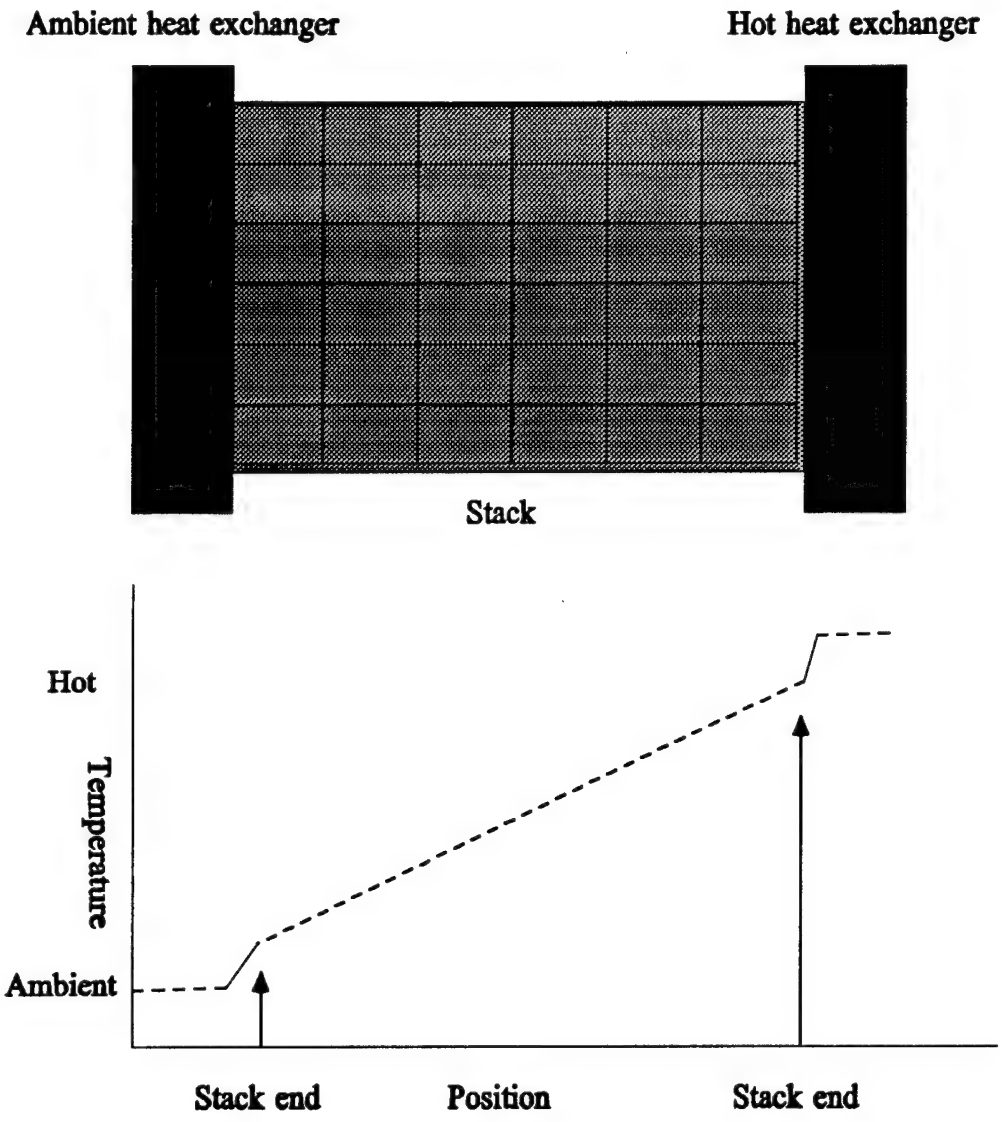

Figure 5.4. The temperature difference between heat exchangers and stack. The temperature of the working fluid in the end of the stack differs from the heat exchanger temperature.

for less than 15% (air) and 12% (helium) of the total losses in this prime mover.

The surface area of the working fluid interacting with the heat exchangers can be controlled by changing the length of the heat exchanger fins. Five sets of 1.90 cm length heat exchangers were built with varying fin lengths. The fin lengths are 0.16, 0.32, 0.64, 1.27, and 1.90 cm. The fin separation and thickness are 0.05 cm in all heat exchangers; these values were chosen due to limitations on commercially available materials.

Theoretical predictions and experimental data for air at atmospheric pressure are shown in Figure 5.5. The experimental onset Delta T_h and Delta T_s differ by approximately 8K for any length of fin and both decrease the same amount with decreasing fin length. This indicates a reduction in attenuation in the prime mover resulting in a lower temperature difference to reach onset. The linear thermoacoustic theory overestimates the losses in the heat exchanger. This becomes more evident as the lengths of the heat exchangers were increased since these losses are proportional to the length.

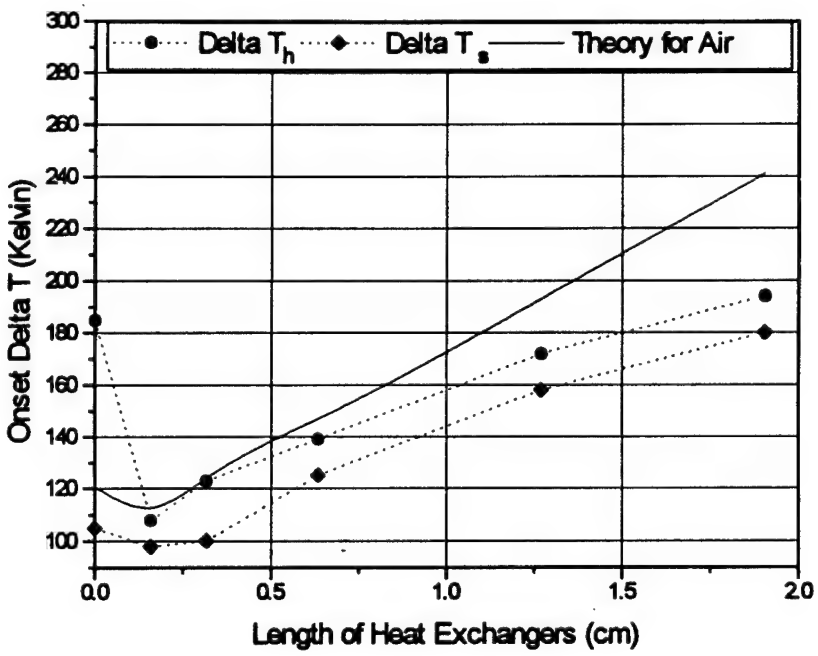

Figure 5.5. The onset Delta T vs. length of the heat exchangers for air at 1 atmosphere. Heat exchanger lengths of 0.15875, 0.3175, 0.635, 1.27 and 1.905 cm are shown. Zero length represents no heat exchangers in the system; the resonator wall is heated and cooled in the appropriate locations. The experimental Delta T_s for the different length heat exchangers is represented by ● and the ■ represents the theoretical values. The ♦ is the temperature difference the prime mover continues to oscillate.

Air was replaced with helium as the working fluid and the experiment was repeated for 1.0, 1.7 and 2.2 atm. Experimental and computed values are shown in Figures 5.6 and 5.7. The agreement is quite good for the 0.159 and 0.317 cm length heat exchangers for all the pressures tested. For longer heat exchangers at 1 atm experiment and theory diverge. The plate separation (R) is approximately equal to the viscous penetration depth at this pressure. The viscous penetration depth is approximately 0.6R for 1.7 and 0.58R for 2.2 atm of helium.

The ambient end of the resonator was then increased to 225 cm. The working fluid was changed to air at 1 atm and the experiment repeated. Results are shown in Figure 5.8. Comparing the onset Delta T_h for the 1.27 cm and 1.905 cm fin length heat exchangers to Delta T_h in Figure 5.5 shows a decrease of 37K and 55K respectively for the longer resonator. The onset temperature for 0.635 cm length heat exchangers was the same for both resonator lengths and increased approximately 10K for the shorter heat exchangers.

Theoretical calculations and experimental data show only modest agreement. The theory overpredicts Delta T for the longer heat exchangers and under predicts Delta T for the shorter heat exchangers. The values happen to coincide at the 1.27 cm length heat exchanger. Referring to Figure 5.8, experimental Delta T for onset increases with length for the short heat exchangers, but reaches a constant value for heat exchangers longer than 0.635 cm.

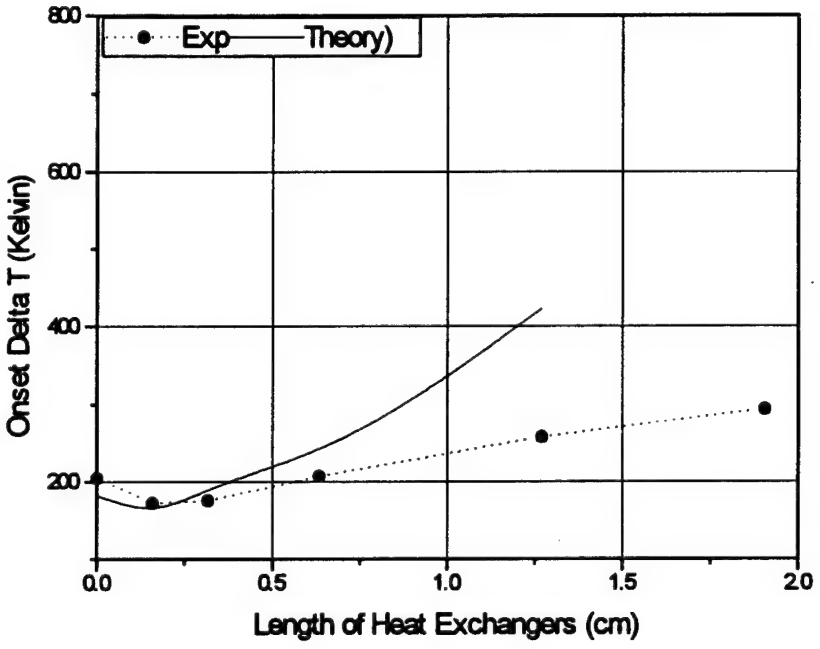

Figure 5.6. The onset Delta T_h vs. length of the heat exchanger for helium at 1 atm.

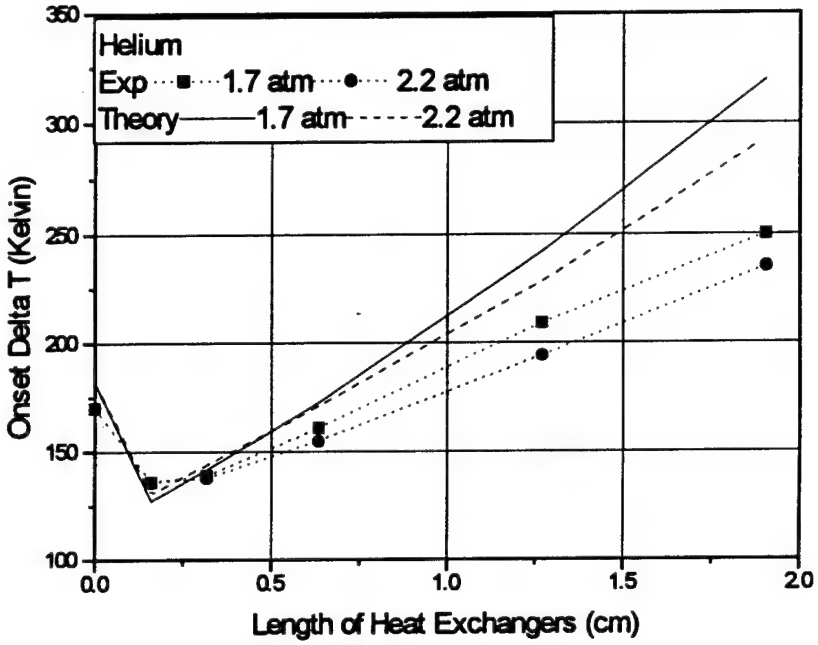

Figure 5.7. The onset Delta T_h vs. length of the heat exchanger for helium at 1.7 and 2.2 atm.

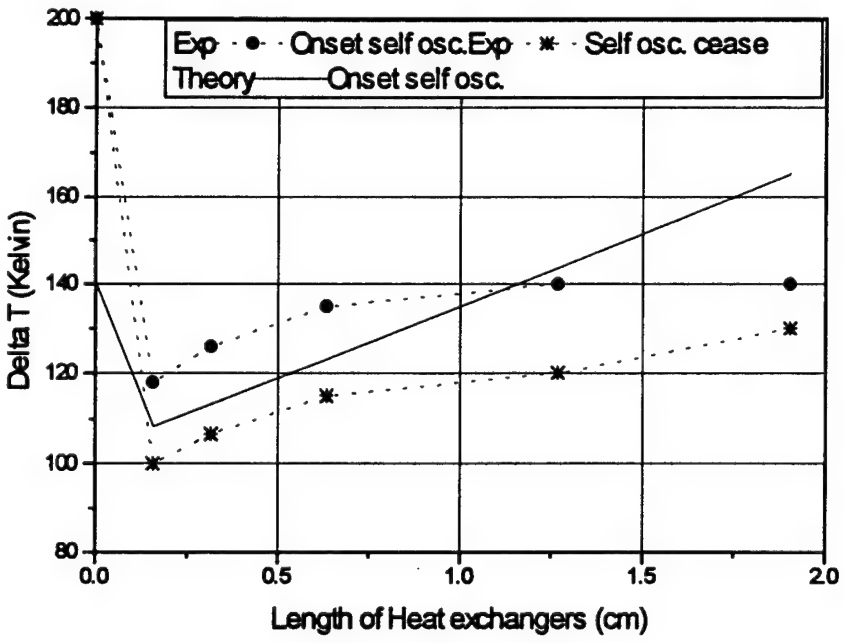

Figure 5.8. The onset Delta T vs. length of the heat exchanger for air at atmospheric pressure and the length of the ambient end of the resonator at 225 cm.

The increase in acoustic losses due to the longer heat exchanger plates affects Delta T less than predicted by theory. The difference is not understood at the present time.

The last set of data presented in Figure 5.8 is the temperature difference necessary to maintain self oscillation after onset is achieved. The heat input was increased until the prime mover began to oscillate. The heat input was then reduced by small amounts. Self oscillation continued until the temperature difference was approximately 20K lower than the temperature difference required to start the oscillations for heat exchanger lengths below 1.25 cm and decreases to approximately 10K for the longest heat exchanger. The Delta T_h necessary to maintain the acoustic oscillation decreases when acoustically stimulated convective heat flow is the dominant heat transfer mechanism.

The zero length heat exchanger represented in Figure 5.8 has no fins. The resonator wall is heated at the hot heat exchanger location and cooled at the ambient heat exchanger location. Theoretically, the heat exchanger was replaced with a cylindrical shell with a plate separation equal to the radius of the resonator. The numerical routine doesn't take this mode of heat transfer into consideration and assumes the temperature of the gas is the same as that of the resonator wall.

5.2 Natural Convection

Figure 5.9a and 5.9b show the two prime movers used in the investigation of the effects of free convection of heat on the onset temperature. Figure 5.9a is the

same configuration used in the Chapter 3 section 4a. The data acquisition and error analysis discussed in Chapter 3 sections 3a and 3b is applied here.

Heat is convected away from the stack in the prime mover depicted in Figure 5.9a and into the stack in the prime mover represented in Figure 5.9b. The convection of heat away from the stack requires a higher temperature difference measured with respect to the heat exchangers for onset of self oscillation. The convection of heat into the stack aids in the transfer of heat to the stack, thus a lower onset temperature difference is observed. The actual temperature difference across the stack was unchanged. The convection of heat away from the stack required a 5K higher temperature difference across the heat exchangers to achieve self oscillation.

Chapter 3 presented the stability curve analysis of a helium filled prime mover. The prime mover used in the stability curve analysis convected heat away from the stack. The difference between experimental and theoretical values was approximately 4K. This difference can now be attributed to free convection of heat away from the stack.

The range of the convective heat flow was investigated by displacing the hot heat exchanger away from the stack and measuring the associated temperature difference required for onset. Figure 5.10 shows the prime mover used in this experiment; it is identical to the prime mover described in Chapter 3 section 2. The onset temperature differences for various gaps are shown in Figure 5.11. The onset

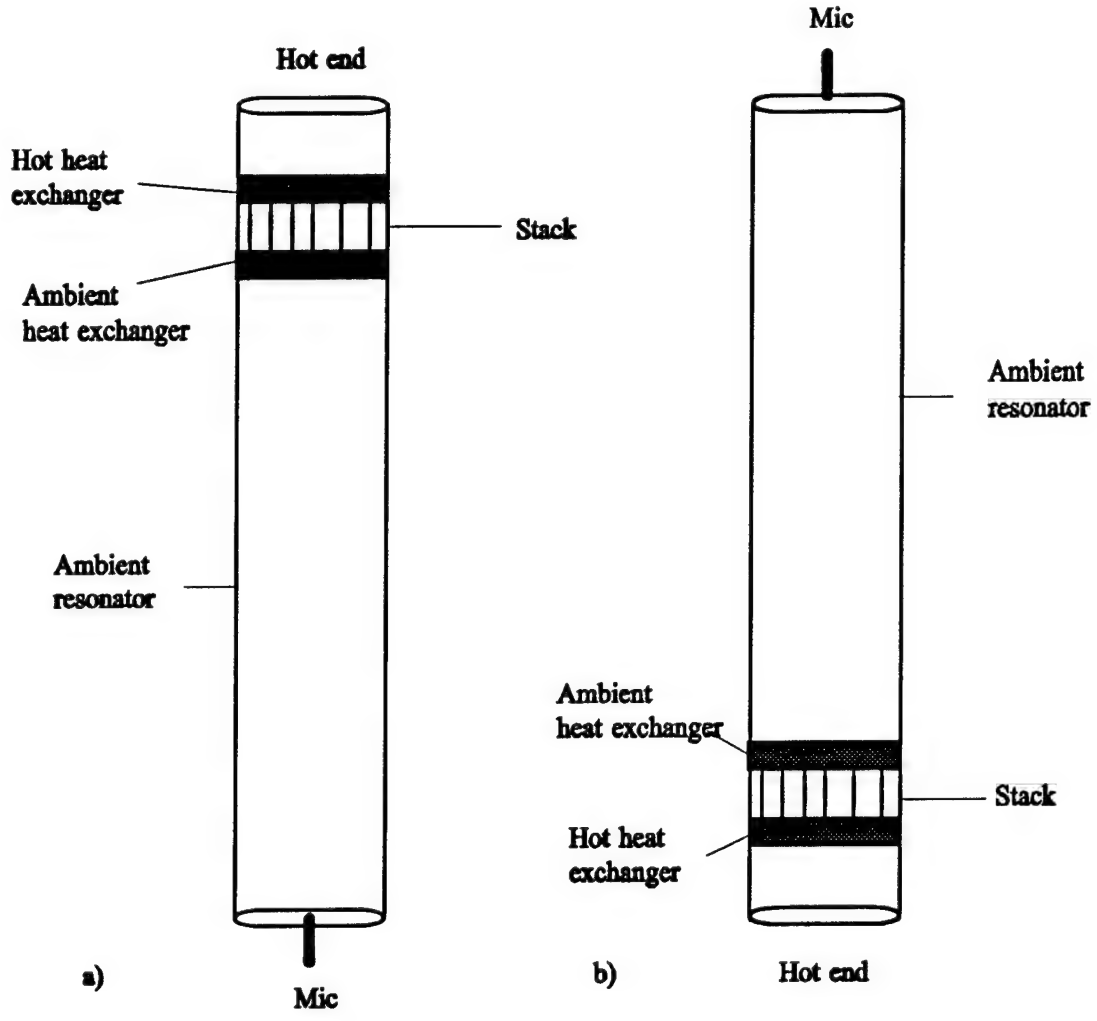

Figure 5.9. Identical prime movers used in the investigation of the convective heat transfer on the onset temperature difference.

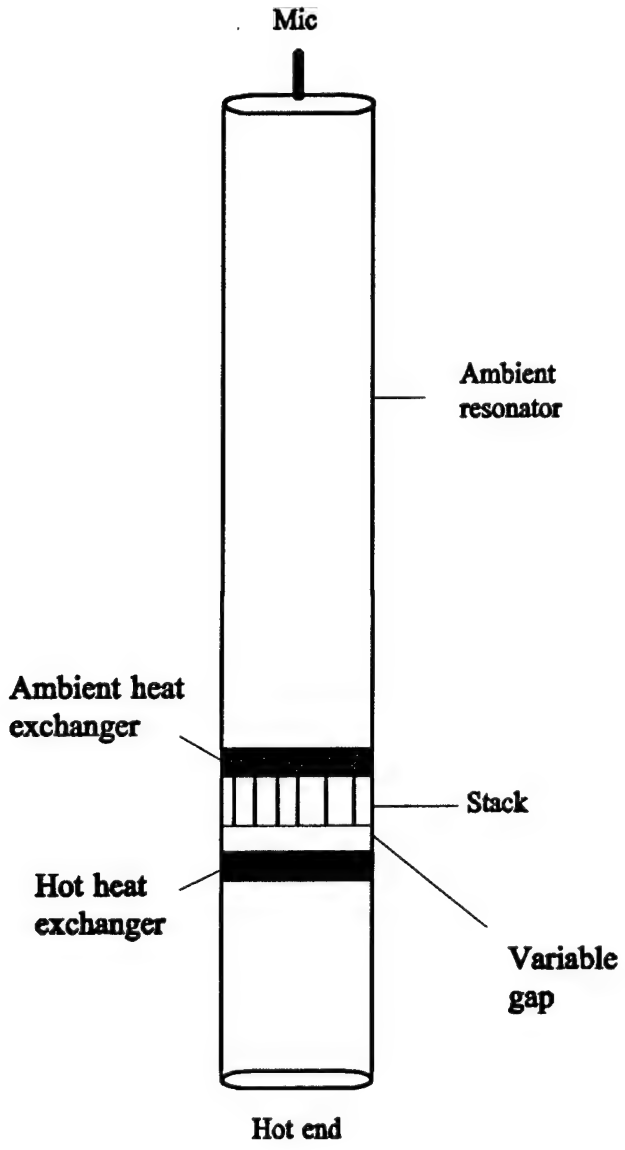

Figure 5.10. The prime mover used to investigate the convective heat transfer. The variable gap allowed the hot heat exchanger to be displaced away from the stack.

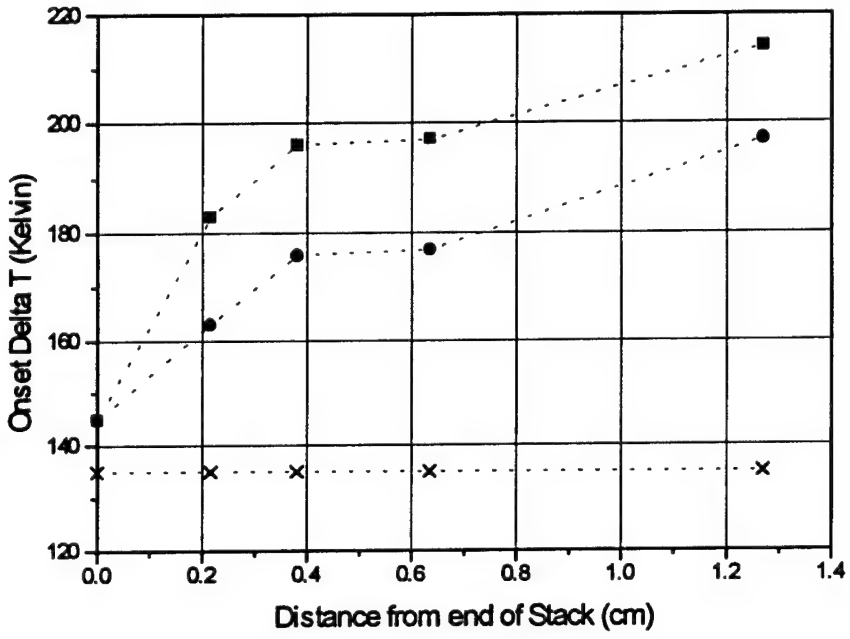

Figure 5.11 The displacement of the hot heat exchanger from the stack. The square data represent the onset temperature difference and the circle data represent the temperature difference at which self oscillation ceases. The data represented by x is the onset temperature difference measured across the ends of the stack and shows the constant Delta T_s necessary for onset of self oscillations.

temperature difference required increases rapidly with small gap length. This demonstrates the short range of convective heat transfer prior to the onset of self oscillations.

The measured temperature difference across the stack was constant for the various gaps tested. The displacement of the hot heat exchanger was toward the pressure antinode. The variations in the viscous and thermal losses due to the displacement were small when compared to the experimental error of $^{+}$/- 1K in the temperature difference measurements.

The reduced thermal energy transfer to the prime mover due to free convection resulted in sustained self oscillations at temperature differences well below that required to start the oscillations as shown in Figure 5.11. This difference was only observed if the heat exchanger was displaced away from the stack. The temperature difference required to maintain self oscillation was approximately 20K lower than the onset temperature difference for any gap. The decrease in temperature difference results from increased convection of heat caused by the oscillatory motion of the working fluid.

The parallel plate heat exchangers commonly used in thermoacoustic devices are heated externally. Heat is then conducted into the thermoacoustic fluid and stack. Large amounts of heat must be supplied to the hot heat exchanger to overcome the

heat lost to conduction to the outside and radiation prior to onset of self oscillations. These external heat losses can be reduced by insulating the heat exchanger.

One way to minimize the external heat loss is to heat the working fluid of the prime mover directly. This can be accomplished by inserting a heating device into the working fluid. Figure 5.12 shows such a heat exchanger. Current is applied to the nichrome wire and the working fluid is heated directly by convection decreasing the amount of heat necessary to reach self oscillation.

The parallel plate heat exchanger requires 300 watts of heat for upwards of 30 minutes to raises the temperature to the point required to achieve onset. The nichrome wire heat exchanger requires 40 watts of heat for less than a minute to achieve the same result. The large temperature difference, approximately 400K, between the gas and the nichrome wire produces a large convective heat flow into the gas. The nichrome wire heat exchanger doesn't rely on a large surface area to supply heat and thermal and viscous losses can be neglected.

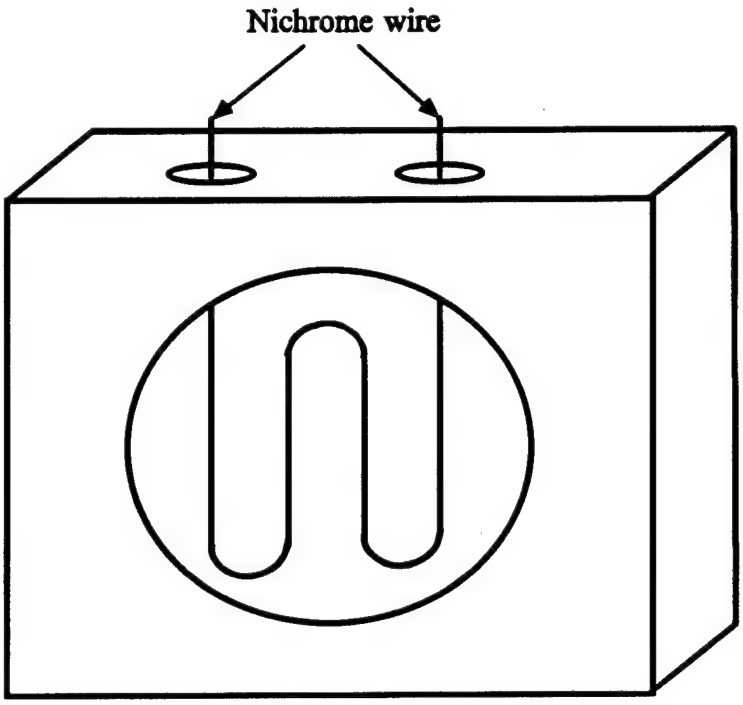

Figure 5.12 Nichrome wire hot heat exchanger. The thermal energy is added directly to the working fluid of the thermoacoustic device.

Chapter 6

Additional elements

6.1 Introduction

The modular design of the thermoacoustic prime mover allows easy placement of additional elements. Additional elements in the resonator give rise to additional losses but can also provide a means of harnessing the work produced by the prime mover. The use of a thermoacoustic prime mover as an acoustic driver for a thermoacoustic heat pump is considered in this chapter. This requires two additional heat exchangers and one additional stack

The introduction of additional elements into a resonator containing a thermoacoustic prime mover increases the viscous and thermal losses requiring a larger Delta T to achieve onset of self oscillation. The losses due to the additional elements depend upon their position in the acoustic standing wave. These losses are a maximum at the velocity antinode at the center of the resonator and should decrease as the additional elements are displaced toward the velocity nodes located at either end of the resonator.

The experimental configuration shown in Figure 6.1 was used to illustrate the dependence of Delta T on the position of an additional element. The data acquisition

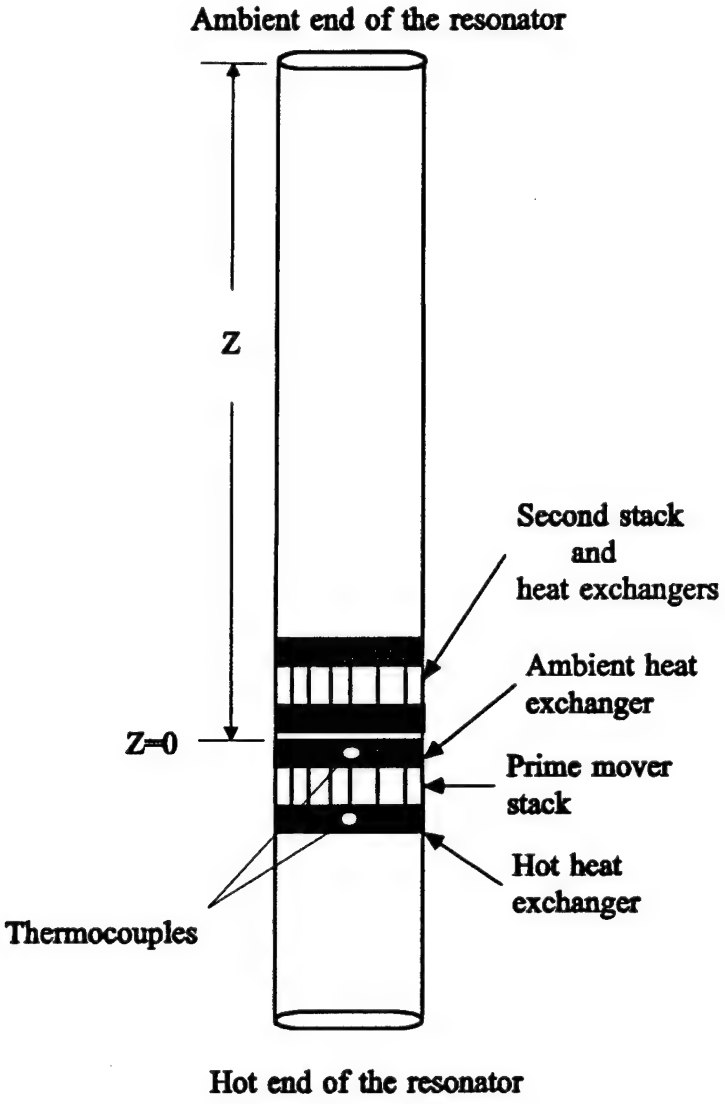

Figure 6.1. Additional elements placed in the prime mover at various distances (Z) from the ambient end of the resonator.

and error analysis is the same as Chapter 3 sections 3a and 3b. The additional stack was 5.08 cm in length and attached too identical 1.9 cm long heat exchangers. The length of the resonator was maintained at 1.62 meters to keep the resonant frequency constant. Air at atmospheric pressure was the working fluid.

The second stack and associated heat exchanger were initially placed in the resonator 1 cm from the prime mover section. Heat was applied to the prime mover until detection of self oscillation. The position of the second stack section and Delta T of the prime mover section were recorded. The system was allowed to cool, the second stack section was moved away from the prime mover section toward the ambient end of the resonator and the experiment repeated.

The experimental onset Delta T for the prime mover with the additional stack-heat exchanger section at various positions is shown in Figure 6.2 and represented by the ■ data. This increase in Delta T as the elements are displaced toward the velocity antinode shows the increase in viscous losses. The short stack approximation was used to theoretically predict the onset Delta T and is represented by the solid line. There is good agreement at all stack locations.

The physical constraints of the resonator limited the positions at which the additional stack-heat exchanger system could be placed. This constraint was removed

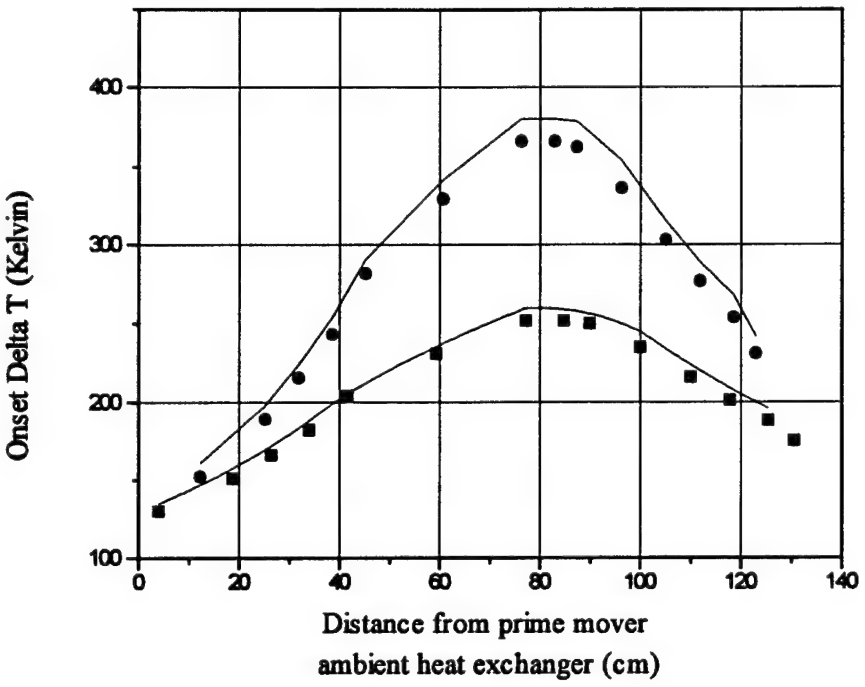

Figure 6.2. The Delta T_h for a prime mover with additional elements at various positions in the resonator. The additional element is located at the ambient end of the resonator at 129 cm and next to the prime mover ambient heat exchanger at zero. The data represented by ● is the Delta T_h for the addition of a identical stack-heat exchanger configuration in the prime mover for air at atmospheric pressure. The data represented by ■ is the Delta T_h for an additional 2.54 cm in length stack only for air at atmospheric pressure. The solid line is the theoretical prediction using the short stack approximation.

by using a 2.54 cm length stack attached to a 0.2 cm diameter rod which was inserted and soldered to the end cap of the ambient end of the resonator. The data represented by ● in Figure 6.2 shows that a maximum Delta T occurs at the velocity antinode and decreases as the stack is positioned toward either end of the resonator. The reduction in length of the second stack section and the removal of the heat exchangers from the movable stack accounts for the reduction in Delta T between sets of data displayed in Figure 6.2. The theoretical prediction is represented by the solid line and is in good agreement with the experimental values.

6.2 Thermoacoustic Heat Driven Refrigerator

The use of a prime mover as a sound source for a heat driven thermoacoustic refrigerator introduces more interactions into the thermoacoustic device. The heat driven refrigerator requires an additional stack, ambient heat exchanger, and cold heat exchanger. The position required to minimize the viscous and thermal losses in the refrigeration stack/heat exchangers is about 1/8 wavelength from the end of the resonator where the prime mover stack/heat exchangers are located. The symmetric position is also available but in that geometry, the cold end of the refrigeration stack faces the ambient end of the prime mover.

The modular design of the thermoacoustic prime mover allows the position of the heat pump section to be changed. Experimentally a 3.0 cm spacer had to be inserted between the heat pump ambient heat exchanger and the prime mover ambient

heat exchanger to generate any temperature difference across the heat pump stack. The reason for this space is not understood at this time but is required in the experimental apparatus. The heat driven refrigerator shown in Figure 6.3 has generated a Delta T of 5 K across the heat pump stack. This is not bad in air at atmospheric pressure or helium at 1.8 atmospheres considering the small surface area of the stack.

6.3 Dual Prime Mover

The addition of an second stack-heat exchanger combination into a resonator containing a prime mover should increase the losses by about a factor of two. The required temperature gradient necessary for onset of self oscillations will increase by a factor of two or more depending on the location of the stack in the resonator.

A dual prime mover system was used to experimentally investigate the temperature distribution and loading caused by a second stack as shown in Figure 6.4. The stacks were 5.08 cm in length. The lengths of the heat exchangers were 1.90 cm. The copper fins of the heat exchangers were 0.05 cm thick and the spacing between adjacent fins was 0.1 cm. The stack-heat exchanger configurations were identical and the short stack approximation was utilized to position the two prime movers in the resonator to minimize losses. The positions were symmetrical with both stack/heat exchanger configurations located a distance of 23 cm from their respective hot end caps.

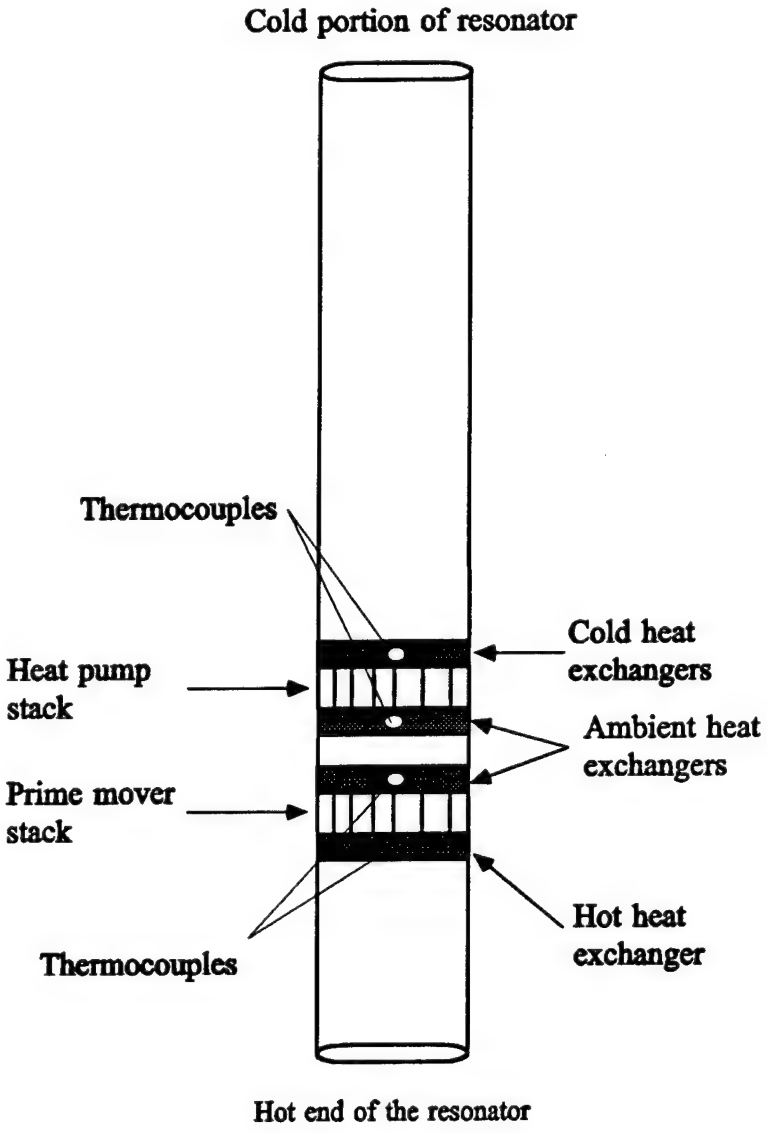

Figure 6.3. Heat driven thermoacoustic refrigerator.

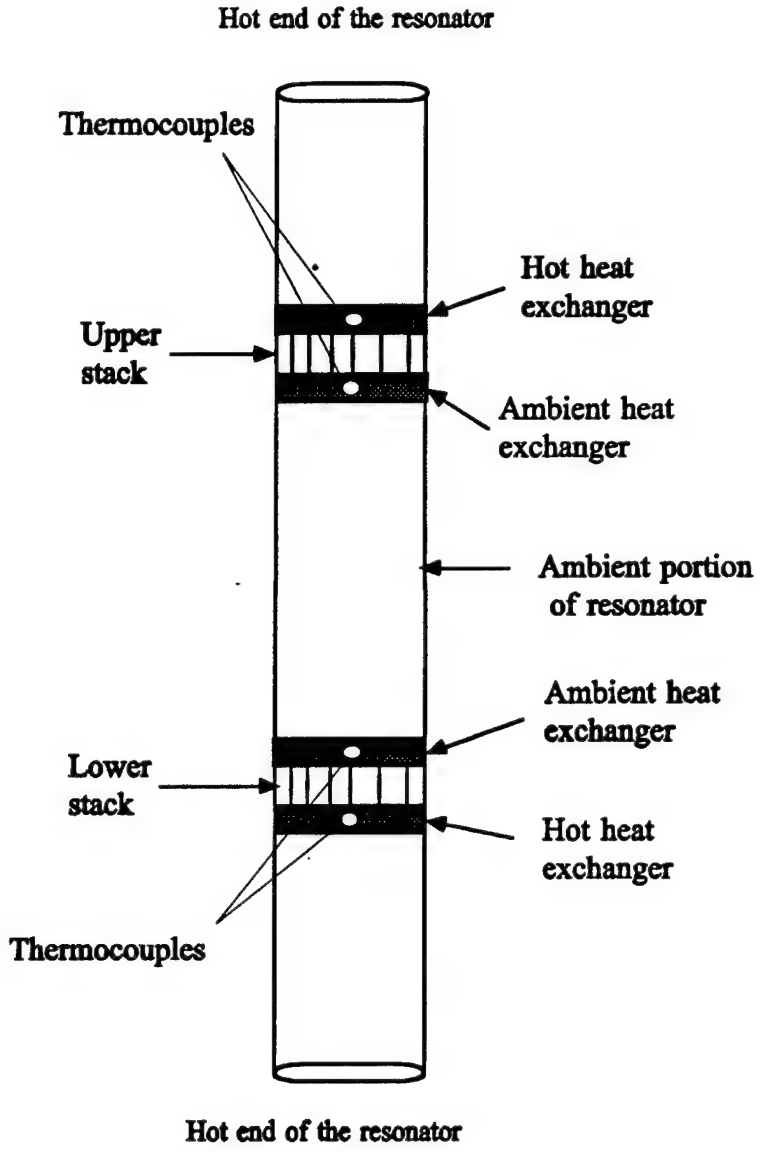

Figure 6.4 Dual Prime mover.

The temperatures of the heat exchangers for the upper prime mover were initially maintained at ambient. Heat was supplied to the lower prime mover until onset of self oscillations was detected. The onset Delta T_h of the lower prime mover and the Delta T_h of the upper prime mover, initially zero, were recorded. The heat source was removed from the lower prime mover and the system was allowed to cool. Delta T_h of the upper prime mover was then increased, heat was again supplied to the lower prime mover, and the experiment was repeated until the Delta T_h for the lower prime mover reached zero.

The experimental results for helium and air are shown in Figure 6.5. The data for air shows that an increase in Delta T for one prime mover leads to an equal decrease in Delta T for the other prime mover. The graph shows a small deviation a straight line for both experiment and theory. This difference is due to variations in temperature dependent gas properties which determine the viscous and thermal penetration depths. The experimental and theoretical values are in good agreement.

The upper prime mover requires a higher Delta T to reach onset when the lower prime mover is isothermal than the lower prime mover when the upper prime mover is isothermal. This is true for both helium and air and is an indication of the role of natural convection aiding heat transfer in the lower prime mover.

The dual heat driven refrigerator is shown in Figure 6.6. The stacks are 5.08 cm in length. The length of the heat exchangers were 1.90 cm for the prime movers

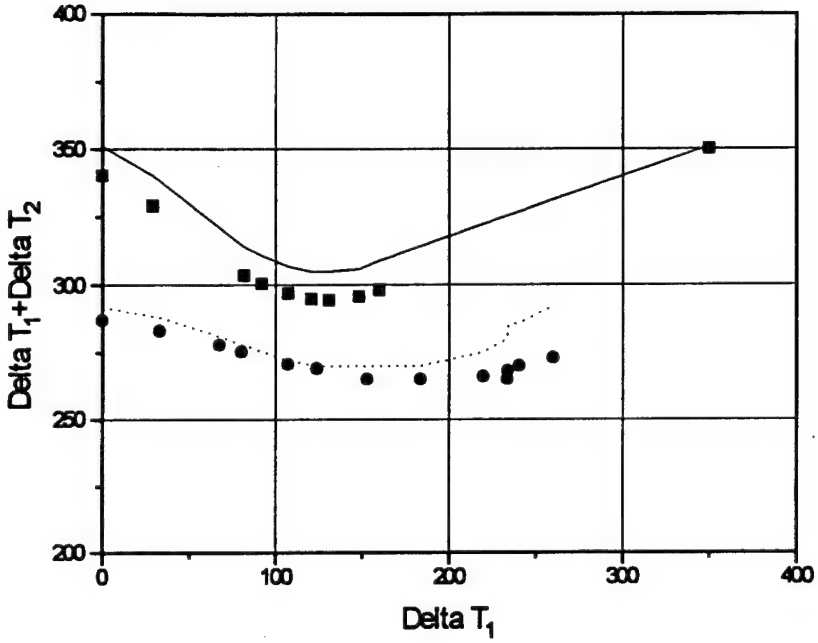

Figure 6.5 The onset temperature difference for a dual prime mover. The data represented by ■ are the onset Delta temperatures for helium at atmospheric pressure. The data represented by ● are the onset Delta temperatures for air at atmospheric pressure. The onset Delta T_h was measured in the heat exchangers. Delta T_1 is for the lower prime mover and Delta T_2 is for the upper prime mover. The solid line is the theoretical prediction for helium and the dotted line is the theoretical prediction for air.

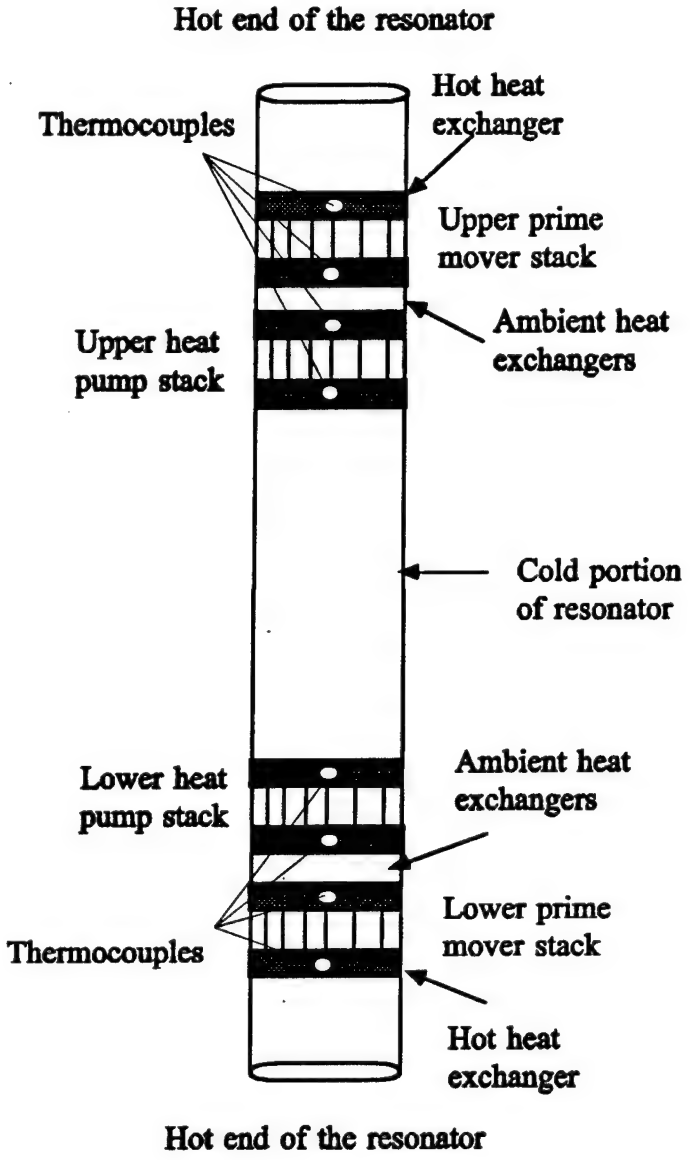

Figure 6.6 Dual heat driven refrigerator

and 0.32 cm for the heat pumps. The copper fins of the heat exchangers were 0.05 cm thick and the spacing between adjacent fins were 0.1 cm for the prime movers and were 0.05 cm in the heat pumps. The stack-heat exchanger configurations were located 23.07 cm from each end of the resonator. The working fluid used was air at atmospheric pressure.

The hot heat exchangers of the two prime movers were heated externally by rod heaters wired in series. The current was increased until onset was detected. The Delta T_h of the prime mover and heat pump sections were measured. The Delta T_h of the lower prime mover section was 320K and the upper prime mover 328K. The heat pumps developed a gradient of 3K. The low temperature difference across the heat pumps can be attributed to low energy density and limited insulation from the room.

Chapter 7

Conclusions

The design and construction of the different thermoacoustic devices described in this dissertation were modular to allow changes in different elements. The primary elements in a prime mover are the ambient end resonator, working fluid, ambient heat exchanger, stack, hot heat exchanger, and hot end resonator.

The onset Delta T of an externally loaded prime mover can be minimized by proper placement of the stack and heat exchangers in the resonator. The short stack approximation, confirmed by experiment, predicted that a minimum Delta T occurs when the stack and heat exchangers are placed approximately 1/8 of a wavelength from the hot end of the resonator. This conclusion is consistent with prior studies.

The thermoacoustic working fluids of choice have been monatomic gases or binary mixtures of monatomic gases. These mixtures and pure gases have high γ and the Prandtl number can be reduced by binary mixing. The monatomic gases helium and argon have N_{pr}=0.67 and γ=1.67 and air has a N_{pr}=0.7 and γ=1.4. The required temperature differences for onset of oscillation are 135K (air), 175K (helium), and 140K (Argon). The diatomic gas (air) and the monatomic gas (argon) have similar onset Delta Ts.

The polyatomic gas sulfur-hexafluride has a $\gamma = 1.04$ and Prandtl number in excess of 0.8. When used as the working fluid in a prime mover sulfur-hexafluride yields a Delta T of 108K which is 62K below the onset Delta T of helium, 27K less than the onset Delta T of air, and 32K less than the Delta T using argon. Polyatomic gas should to be considered as working fluids in thermoacoustics.

The working fluid interactions in a thermoacoustic device may be optimized by using binary gas mixtures. The thermodynamic properties may be adjusted to maximize the thermodynamic processes while minimizing the viscous losses. The minimum temperature difference necessary for the onset of self oscillation for a binary gas mixture occurs when the working fluid is composed of 60% of the lighter species and 40% of the heavier species. The binary gas mixture containing 60% helium and 40% sulfur hexafluride has a $N_{pr} \approx 0.24$ and $\gamma \approx 1.2$. The onset Delta T of a prime mover using this working fluid is 77K. The binary gas mixture of 60% helium and 40 argon has a $N_{pr} \approx 0.42$ and $\gamma \approx 1.66$ and yields an onset Delta T of 120K.

The binary mixture of air-neon, air-argon, and air-krypton, in any percentage had no effect on the computed Prandtl number while the binary mixture of air-helium decreased the Prandtl number by approximately 30%. For binary gas mixtures to be used in thermoacoustics the mass ratio for the two species must be at least five to have a noticeable effect on the Prandtl number.

The viscous and thermal losses are shown to be directly proportional and the acoustic gain inversely proportional to the length of the stack for a given temperature difference across the stack. The onset Delta T_s is defined as the temperature difference when the gain in the stack becomes larger than the total losses of the prime mover. The onset Delta T_s can be decreased for a prime mover by decreasing the length of the stack. The reduction in the stack length is limited by the conduction of heat down the stack holder, stack, and gas which reduces the ability to maintain a temperature difference at the ends of the stack. The theoretical prediction sets no limit on the length of the stack. The theory assumes any heat conducted down the stack can be expelled by the ambient heat exchanger maintaining the linear temperature gradient.

Heat exchangers are used in thermoacoustic devices to supply and extract heat at the ends of the stack to establish and maintain a temperature gradient. Their presence in a thermoacoustic device are purely dissipative in both viscous and thermal processes. The viscous and thermal losses are proportional to the length of the heat exchangers. Decreasing the length of the heat exchangers decreases the total acoustic losses in the prime mover and the required Delta T_s. Delta T_s is the actual temperature gradient necessary across the stack to sustain acoustic oscillations. Delta T_s is always smaller than predicted. This is due to the assumption that the gas entering the stack end is the same temperature as the gas contained in the heat exchanger. This

compares experimentally to Delta T_h; the temperature difference between heat exchangers.

Losses in the heat exchangers can be reduced by decreasing the lengths of the heat exchangers, thus decreasing the onset Delta T_s required to reach onset. The shortest heat exchanger tested gives a Delta T_s lower than observed with no heat exchanger. This suggests that heat exchanger losses can be effectively reduced by decreasing length but some fins are necessary to insure proper heat transfer to the working fluid.

Linear thermoacoustic theory over predicts losses in the heat exchangers for fin separations of 0.05 cm which is less than twice the viscous penetration depth for helium or air at 1 atm. In Chapter 3 the heat exchangers used had a plate separation of 0.1 cm which is greater than twice the viscous penetration depth for helium or air at 1 atm. In this case, theoretical values agreed quite well with the experimental values for a 1.9 cm long heat exchanger. There was good agreement between theory and experiment for short heat exchangers. The short heat exchangers have little viscous losses so an error in that term would not be as important to predictions.

The transfer of heat by natural convection was shown experimentally to increase the efficiency of the parallel plate heat exchangers. The hot end resonator and hot heat exchanger must be below the stack to take full advantage of this mode of heat transfer. The Delta T_h required for onset of self oscillation of a prime mover is 5K

lower taking advantage of natural convective heat flow; Delta T_s remains constant. The stability analysis data present in Chapter 3 section 4a contained a 4K difference between experiment and theory at all pressures tested. This can now be attributed to natural convection.

Convective heat transfer is shown to be short range in the conventional parallel plate heat exchanger. Convection must transport gas into the stack to aid in heat transfer from the heat exchanger to the stack. This was experimentally demonstrated by moving the hot heat exchanger away from the stack. The onset Delta T_h increased 50K for a displacement of 0.32 cm and increases 80K for a displacement of 1.27 cm; the onset Delta T_s remains constant for all stack/heat exchanger separations. The Delta T_h necessary to start the oscillations was 20K higher than the Delta T_h required to maintain the oscillations when the stack and heat exchanger were separated. The acoustic oscillations stimulate convective heat flow into the stack that bridges the gap increasing the efficiency of the heat exchangers.

Acoustically stimulated convection is proportion to the particle displacement, volume of gas contained in the heat exchanger and the resonant frequency. This forced convection was only observed if the prime mover element placements were other than optimum for a minimum Delta T. Increasing the length of the ambient end of the resonator shifted the stack/heat exchangers to 1/11 of a wavelength from the

hot end cap and produced the same decrease in optimum Delta T as displacing the hot heat exchanger away from the stack.

The parallel plate heat exchangers used in these experiments are very inefficient and require the application of large amounts of heat to raise the temperature of the gas in thermal contact with the hot heat exchanger. Heat lost to the surroundings due to conduction, convection, and radiation reduces the amount of heat reaching the working fluid at the end of the stack. The direct application of heat to the working fluid by a nichrome wire decreases the heat lost to the surroundings. The use of nichrome wire heat exchanger also reduces the thermal energy required to initially reach Delta T and produce acoustic oscillations.

The heat driven refrigerator produced very little cooling with air at ambient pressure and helium at 1.8 atmospheres as the working fluids. The cold end of the resonator was exposed to the room which acted as a large load on the heat pump section. The prime mover was designed to produce oscillations at the lowest possible temperature. The acoustic amplitudes produced, in most cases was less than 3%.

The onset Delta T_s of a 5 cm long stack prime mover with a binary gas mixture of 60% helium and 40% SF_6 using the smallest length heat exchangers and stack can be reduced to 55 K. The utilization of natural convection can decrease this by another 5K. It should be possible to build a heat driven thermoacoustic refrigerator that requires a Delta T_s of approximately 100 K which is often available as waste heat.

However the acoustic amplitudes, in the range of 5-10% ambient pressure, necessary for useful cooling by the heat pump section could elevate this Delta T to 135-175K.

Future Work

The low pressure amplitudes of the minimum Delta T prime movers should be increased if the device is to be considered in a heat driven refrigerator. This can be done by increasing the cross sectional area of the stack, modifying the working fluid or slight deviation from a minimum Delta T prime mover.

The working fluids of choice in thermoacoustics have been monatomic or binary mixtures of monatomic gases. This needs to be extended to include polyatomic gas such as sulfur-hexafluride and carbon dioxide. The pressure amplitudes for polyatomic gases need to be investigated and compared to the working fluids used in thermoacoustic devices. Efficiencies and coefficient of performance must be investigated to determine if they will be useful in thermoacoustics.

The acoustically stimulated heat flow generated in a non-optimized system should be investigated. This heat flow reduced the Delta T once the prime mover was oscillating by 20K in all cases where it was observed. A prime mover designed to operate at low Delta T could be modified slightly to operate at a slightly higher Delta T. This could possibly harness the stimulated heat flow to lower the temperature after onset and lowering the actual Delta T by 10% or more.

LIST OF REFERENCES

References

1. C. Sondhauss, "Üeber die Schallchwingungen der luft in erhitzter Glasrohren und in gedeckten pferfen von ungleicher Weite," Ann. Phys. (Leipzig) **79**,1 (1850).

2. Baron Rayleigh, J. W. Strutt, The Theory of Sound (Dover,New York, 1945, 2nd ed., vol 2 sec. 322

3. G. Kirchhoff, "Ueber den Einfluss der Warmeleitung in einem Gas auf die Schallbewegung," Ann. Phys.**134**, 177 (1868)

4. K. W. Taconis, "Vapor-liquid equilibrium of solutions of ^{3}He in ^{4}He," Physica **15**, 738 (1949).

5. A. Kramers, "Vibrations of a gas column," Physica **15**, 971-984 (1949)

6. N. Rott, "Damped and thermally driven acoustic oscillations in wide and narrow tubes," Z. Angew Math Physics **20**, 230-243 (1969)

7. N. Rott, "Thermally driven acoustic oscillation Part II. Stability limit for Helium," Z. Angew Math Physics **24**, 54-72 (1973)

8. N. Rott, "The influence of heat conduction on acoustic streaming," Z. Angew Math Physics **25**, 417-421 (1974a)

9. N. Rott, "The heating effect connected with non-linear oscillations in a resonance tube," Z. Angew Math Physics **25**, 619-634 (1974b)

10. N. Rott, "Thermally driven acoustic oscillations Part III Second-order heat flux," Z. Angew. Math Physics **26** 43-49 (1975)

11. N. Rott and G. Zouzoulas, "Thermally driven acoustic oscillations, part IV: tubes with variable cross-section," Z. Angew. Math Physics **34**, 609 (1976)

12. T. Yazaki, A. Tominaga, and Y. Narahara (1979). Stability limit for thermally driven acoustic oscillations. Cryogenics **19**, 393-396.

13. P. Merkli and H. Thomann, "Thermoacoustic effects in a resonance tube," J. Fluid Mech. **70**, 161 (1975).

14. Wheatley J. C., Hofler T., Swift G. W., and Migliori A., "An intrinsically irreversible thermoacoustic heat engine," J. Acoust. Soc. Am. **74**, 153 (1983)

15. T. J. Hofler, "Thermoacoustic refrigerator design and performance," Ph. D. dissertation, Physics Department, University of California at San Diego (1986)

16. G. W. Swift, "Thermoacoustic engines," J. Acoust. Soc. Am. **84**, 1145-1180 (1988)

17. A. A. Atchley, H.E. Bass, T. J. Hofler, and H. T. Lin, "Study of a thermoacoustic prime mover below onset of self oscillation," J. Acoust. Am. **91**. 734-743 (1992)

18. A. A. Atchley, "Standing wave analysis of a thermoacoustic prime mover below onset of self oscillation," J. Acoust. Soc. Am. **92**(5) 2907-2914 (1992)

19. W. P. Arnott, Henry E. Bass, and Richard Raspet, "General formulation of thermoacoustics for stacks having arbitrarily-shaped pore cross-sections," J. Acoust. Soc. Am. **91**, 3228 (1991)

20. R. Raspet, H.E. Bass, and J. N. Kordomenos, "Thermoacoustics of traveling waves: Theoretical analysis for an ideal inviscid gas," J. Acoust. Soc. Am. **94**(3) 2232-2239 (1993)

21. J. Kordomenos, A. A. Atchley, R. Raspet, and H. E. Bass, "Experimental study of a thermoacoustic termination of a traveling wave," J. Acoust. Soc. Am. **98**(3), 1623-1628 (1995)

22. P. Arnott, J. A. Lightfoot, R. Raspet, and H. Moosmüller,"Radial wave thermoacoustic engines; Theory and examples for refrigerators prime movers and high-gain narrow-bandwidth photoacoustic spectrometers," J. Acoustic Soc. Am. **99**(2), (1996)

23. J. A. Lightfoot, "Analysis of a Radial Wave prime mover with Variable Stack Spacing," Prospectus for a Doctoral Dissertation, Department of Physics and Astronomy, University of Mississippi, (1994)

24. B. Ward and G Swift,"Design Environment for Low-Amplitude Thermoacoustic Engines," Los Alamos National Laboratory, (1994)

25. W. P. Arnott, Numerical routine based on the "General formulation of thermoacoustics for stacks having arbitrarily-shaped pore cross-sections," (1991)

26. W. Patrick Arnott, James R. Belcher, Richard Raspet, and Henry E. Bass, "Stability analysis of a helium filled thermoacoustic engine,"J. Acoust. Soc. Am. **96**(1), 370-375 (1994)

27. F. Reif, Fundamentals of Statistical and Themal Physics, McGraw-Hill, Inc. 1965

28. J. Hirschfelder, C. Curtiss, and R. Bird, Molecular Theory of Gases and Liquids, 2nd Edition, John Wiley & Sons, Inc., New York 1964

29 S. Chapman and T. Cowling, The Mathematical Theory of Non-uniform Gases, 3rd Edition, Cambridge at the University Press, 1970

30. A. Eucken, Physik. Z., **14**, 1913

31. D. Enskog, "Kinetische Theorie der Vorgänge in mäßig verdünnter Gasen Archiv. for Matematik, och Fysik,**16**,1917

32. Matheson Gas Data Book, Fifth Edition, 1971

33. G. W. Swift, "Analysis and performance of a large thermoacoustic engine," J. Acoust. Soc. Am. **92** 1551-1563 (1992)

34. M. Stinson, "The propagation of plane waves in narrrow and wid circular tubes, and generalization to uniform tubes o aribitrary cross-sectional shape," J. Acoust. Soc. Am. **89**, 550-558 (1991)

35. A. D. Pierce, Acoustics: An Introduction to Its Physical Principles and Applications (Acoustical Society of America, New York, 1989), Chapter 10.

36. H. Roh, "Measurement and calculation of acoustic propagation constants in arrays of small air-filled rectangular tubes," J. Acoust. Soc. Am. **89** (6), 2617-2624 (1991)

Appendix

Appendix A

Arnott's program for thermoacoustic prime mover. The use of a fourth order Runge Kutta integration to calculate the impedance and pressure across the stack for various pore geometries to minimize the onset Delta T. The short stack approximation is utilized.

```
      Program Engine
      Implicit double precision (a-h,o-z)
      Character*30 fname
      Call system('cls')
      Call system('dir /b *.par')
      Write(*,*) 'Enter the input file :    '
      Read(*,1093) fname
      Call system('cls')
      Write(*,*) 'Calculating for the file : ', fname
      Write(*,*)
      Write(*,*)
1093  Format(a30)
      Open(2,file=fname)
! Obtain the input parameter file.
      Call setval
! Evaluate the input parameter file.
      Call scan
      Close(2)
      End
```

```
Descriptor for this file.
Second line of file descriptor.
----------*HC*STACK*HC*-----------------------------------------------------|
|         *HC*STACK*HC*                                                     |
|         *HC*STACK*HC*                                                     |
|         *HC*STACK*HC*                                                     |
|         *HC*STACK*HC*                                                     |
----------*HC*STACK*HC*-----------------------------------------------------|
TARGETS:  The hot end temperature can either be near the rigid wall, or
near the pressure antinode at the tube center. The temperature of the
right and left ends of the tube near the rigid wall is set to 0.d0 if
it is to be the target.  Ambient temperature becomes that of the tube
nearest the center. REVERSE THIS ARRANGMENT TO MAKE THE CENTER THE TARGET.
TO SIMPLY GET THE QUALITY FACTOR, HAVE THE TEMPERATURE NON ZERO EVERYWHERE.
The resonator is assumed to be symmetric about the center.
User inputs the length of each section along with section type
and element type.  Some inputs are now obsolete as noted by OBS at the end
of the line. This version works for one dimensional
resonators. Runge Kutta integration is now used in all thermoacoustic
elements, not only the stack. These values are used in setvalues.f
WORKING FLUID: CHOICES ARE AIR, HELIUM, HE60AR40 He-Ar mixture.
HELIUM
WORKING STACK SOLID: CHOICES ARE STAINLESS, KAPTON, POLYIMID, COPPER.
KAPTON
TERMINATION AT THE RIGHT END OF THE TUBE.
ONE OF RIGID, FREE, OR INFIN.  INFIN IS AN INFINITE TUBE.
        RIGID
TERMINATION AT THE LEFT END OF THE TUBE.
ONE OF RIGID, FREE, NODE OR INFIN.INFIN=INFINITE_TUBE.NODE=velocity_node.
        RIGID
AMBIENT PRESSURE IN THE TUBE AND THE PRESSURE AMPLITUDE at the right end
where both numbers are given in Pascal.
  1.8d5   1.000000000000000
NUMBER OF SECTIONS IN THE TAE.  E.G. AN OPEN SECTION, RGH HEAT EXCH,
STACK, LEFT HEAT EXCH, AND ANOTHER OPEN SECTION WOULD BE 5.  INTEGER.
          5    Automatically assumed in this program.
******** DEFINITION OF SECTION 5 ************************************
SECTION TYPE, ONE OF OPENTU, HEXCH, OR STACK.
          OPENTU
Heat load on the element in case it is a heat exchanger.
Give the number in Watts.
  0.d0          Not currently used.
NUMBER OF LAYERS THIS SECTION IS BROKEN UP INTO.
1<= NUMLAY <= 100 PRACTICALLY.
          1
LENGTH OF THE SECTION.
METERS.
          23.07d-2
```

```
TEMPERATURE OF THE RGH END OF THE SECTION.  FOR AN ISOTHERMAL SECTION
SUCH AS OPEN TUBE OR HEAT EXCHANGERS, USE TRGH = TLEFT.  KELVIN.
0.d0        ENTER 0.d0 if the wall end is a target.
TEMPERATURE AT THE LEFT END OF THE SECTION.
SEE NOTE ABOVE.  KELVIN.
0.d0        ENTER 0.d0 if the wall end is a target.
RATIO OF 2 PORE AREA TO PORE PERIMETER (M). FOR: CYL=RADIUS, SLIT=
WIDTH, RECT=2SW A/(1+A) A>1=SIDES ASPECT RATIO, SW=SHORTEST SEMIWIDTH.
          4.275D-2
ASPECT RATIO OF THE PORE. VALID FOR RECTANGULAR PORES ONLY.
NECESSARY AS A GENERAL RULE.
          1.0D0
POROSITY OF THE SECTION.  FOR OPEN TUBE, USE POROSITY = 1.
FOR OTHER TYPES OF SECTIONS, POROSITY <=1.
          1.D0
END OF SECTION 5. ************************************************************
******** DEFINITION OF SECTION 4 ***************************************
SECTION TYPE, ONE OF OPENTU, HEXCH, OR STACK.
          HEXCH
Heat load on the element in case it is a heat exchanger.
Give the number in Watts.
  467.68298339844
NUMBER OF LAYERS THIS SECTION IS BROKEN UP INTO.
1<= NUMLAY <= 100 PRACTICALLY.
          1
LENGTH OF THE SECTION.
METERS.
          1.762D-2
TEMPERATURE OF THE RGH END OF THE SECTION.  FOR AN ISOTHERMAL SECTION
SUCH AS OPEN TUBE OR HEAT EXCHANGERS, USE TRGH = TLEFT.  KELVIN.
  450.
TEMPERATURE AT THE LEFT END OF THE SECTION.
SEE NOTE ABOVE.  KELVIN.
  450.
RATIO OF 2 PORE AREA TO PORE PERIMETER (M). FOR: CYL=RADIUS, SLIT=
WIDTH, RECT=2SW A/(1+A) A>1=SIDES ASPECT RATIO, SW=SHORTEST SEMIWIDTH.
                1.016D-3
ASPECT RATIO OF THE PORE. VALID FOR RECTANGULAR PORES ONLY.
NECESSARY AS A GENERAL RULE. ASPECT RATIO => 1 ALWAYS.
          1.0D0
POROSITY OF THE SECTION.  FOR OPEN TUBE, USE POROSITY = 1.
FOR OTHER TYPES OF SECTIONS, POROSITY <=1.
          .61D0
END OF SECTION 4. ************************************************************
******** DEFINITION OF SECTION 3 ***************************************
SECTION TYPE, ONE OF OPENTU, HEXCH, OR STACK.
          STACK
Heat load on the element in case it is a heat exchanger.
```

```
Give the number in Watts.
   0.d0
NUMBER OF LAYERS THIS SECTION IS BROKEN UP INTO.
1<= NUMLAY <= 100 PRACTICALLY. (WAS 20 AT ONE TIME)
          15
LENGTH OF THE SECTION.
METERS.
                  5.08D-2
TEMPERATURE OF THE RGH END OF THE SECTION.  FOR AN ISOTHERMAL SECTION
SUCH AS OPEN TUBE OR HEAT EXCHANGERS, USE TRGH = TLEFT.  KELVIN.
  450.
TEMPERATURE AT THE LEFT END OF THE SECTION.
SEE NOTE ABOVE.  KELVIN.
  293.
RATIO OF 2 PORE AREA TO PORE PERIMETER (M). FOR: CYL=RADIUS, SLIT=
WIDTH, RECT=2SW A/(1+A) A>1=SIDES ASPECT RATIO, SW=SHORTEST SEMIWIDTH.
                  0.77D-3
ASPECT RATIO OF THE PORE. VALID FOR RECTANGULAR PORES ONLY.
NECESSARY AS A GENERAL RULE. ASPECT RATIO => 1 ALWAYS.
          1.0D0
POROSITY OF THE SECTION.  FOR OPEN TUBE, USE POROSITY = 1.
FOR OTHER TYPES OF SECTIONS, POROSITY <=1.
          .69D0
END OF SECTION 3. ************************************************************
******** DEFINITION OF SECTION 2 ***************************************
SECTION TYPE, ONE OF OPENTU, HEXCH, OR STACK.
          HEXCH
Heat load on the element in case it is a heat exchanger.
Give the number in Watts.
   10.d0
NUMBER OF LAYERS THIS SECTION IS BROKEN UP INTO.
1<= NUMLAY <= 100 PRACTICALLY.
          1
LENGTH OF THE SECTION.
METERS.
          1.643D-2
TEMPERATURE OF THE RGH END OF THE SECTION.  FOR AN ISOTHERMAL SECTION
SUCH AS OPEN TUBE OR HEAT EXCHANGERS, USE TRGH = TLEFT.  KELVIN.
          293.
TEMPERATURE AT THE LEFT END OF THE SECTION.
SEE NOTE ABOVE.  KELVIN.
          293.
RATIO OF 2 PORE AREA TO PORE PERIMETER (M). FOR: CYL=RADIUS, SLIT=
WIDTH, RECT=2SW A/(1+A) A>1=SIDES ASPECT RATIO, SW=SHORTEST SEMIWIDTH.
                  1.016d-3
ASPECT RATIO OF THE PORE. VALID FOR RECTANGULAR PORES ONLY.
NECESSARY AS A GENERAL RULE. ASPECT RATIO => 1 ALWAYS.
          1.0D0
```

```
POROSITY OF THE SECTION.  FOR OPEN TUBE, USE POROSITY = 1.
FOR OTHER TYPES OF SECTIONS, POROSITY <=1.
          .53D0
END OF SECTION 2 ************************************************************
******** DEFINITION OF SECTION 1 ***************************************
SECTION TYPE, ONE OF OPENTU, HEXCH, OR STACK.
          OPENTU
Heat load on the element in case it is a heat exchanger.
Give the number in Watts.
   0.d0
NUMBER OF LAYERS THIS SECTION IS BROKEN UP INTO.
1<= NUMLAY <= 100 PRACTICALLY.
          1
LENGTH OF THE SECTION.
METERS.
          129.d-2
TEMPERATURE OF THE RGH END OF THE SECTION.  FOR AN ISOTHERMAL SECTION
SUCH AS OPEN TUBE OR HEAT EXCHANGERS, USE TRGH = TLEFT.  KELVIN.
293.d0          ENTER 0.d0 if the center is a target.
TEMPERATURE AT THE LEFT END OF THE SECTION.
SEE NOTE ABOVE.  KELVIN.
293.d0          ENTER 0.d0 if the center is a target.
RATIO OF 2 PORE AREA TO PORE PERIMETER (M). FOR: CYL=RADIUS, SLIT=
WIDTH, RECT=2SW A/(1+A) A>1=SIDES ASPECT RATIO, SW=SHORTEST SEMIWIDTH.
          4.258D-2
ASPECT RATIO OF THE PORE. VALID FOR RECTANGULAR PORES ONLY.
NECESSARY AS A GENERAL RULE. ASPECT RATIO => 1 ALWAYS.
          1.0D0
POROSITY OF THE SECTION.  FOR OPEN TUBE, USE POROSITY = 1.
FOR OTHER TYPES OF SECTIONS, POROSITY <=1.
          0.992D0
END OF SECTION 1. ************************************************************

************************************************************************
**SUBROUTINE SETVAL * GET THE INPUT PARAMETERS FROM AN EXTERNAL FILE**
************************************************************************
    SUBROUTINE SETVAL
        implicit double precision (a-h,o-z)
* VARIABLES WHICH DEFINE THE TAE.
    CHARACTER*70 SECTYP(100),ETYPE(100),TERMINR,TERMINL
    INTEGER NUMLAY(100),NUMSEC
    double precision DELEM(100),TRGH(100),TLEFT(100),RATIO(100)
   *,ASPECT(100),POROS(100),HEATLOAD(100)
    INTEGER J
    CHARACTER DUMMY
        character*70 config,geometry,fluid,solid
        common /switchs/ geometry,fluid,solid,config
    COMMON /VARS1/ SECTYP,ETYPE,TERMINR,TERMINL
```

```
      COMMON /VARS2/ NUMLAY,NUMSEC
      COMMON /VARS3/ DELEM,HEATLOAD,TRGH,TLEFT,RATIO,
     *  ASPECT,POROS,PAMB,P1not
1     FORMAT (A1)
2     FORMAT (A70)
      REWIND 2
          do 300 j=1,24
300       READ (2,1) DUMMY
      geometry='onedim'
                  read (2,2) fluid
      CALL NOPAD(fluid)
      READ (2,1) DUMMY
                  read (2,2) solid
      CALL NOPAD(solid)
      READ (2,1) DUMMY
      READ (2,1) DUMMY
                  READ (2,2) TERMINR
      CALL NOPAD(terminr)
      READ (2,1) DUMMY
      READ (2,1) DUMMY
                  READ (2,2) TERMINL
      CALL NOPAD(TERMINL)
      READ (2,1) DUMMY
      READ (2,1) DUMMY
                  READ (2,*) PAMB,P1not
      READ (2,1) DUMMY
      READ (2,1) DUMMY
                  READ (2,*) NUMSEC
          dtotal=0.d0
                              DO 10 J=NUMSEC,1,-1
      READ (2,1) DUMMY
      READ (2,1) DUMMY
                  READ (2,2) SECTYP(J)
      CALL NOPAD(SECTYP(J))
      READ (2,1) DUMMY
      READ (2,1) DUMMY
                  READ (2,*) HEATLOAD(J)
          ETYPE(J) ='Rect'
      READ (2,1) DUMMY
      READ (2,1) DUMMY
                  READ (2,*) NUMLAY(J)
      READ (2,1) DUMMY
      READ (2,1) DUMMY
                  READ (2,*) DELEM(J)
                  dtotal = dtotal + delem(j)
      READ (2,1) DUMMY
      READ (2,1) DUMMY
                  READ (2,*) TRGH(J)
```

```
      READ (2,1) DUMMY
      READ (2,1) DUMMY
                 READ (2,*) TLEFT(J)
      READ (2,1) DUMMY
      READ (2,1) DUMMY
                 READ (2,*) RATIO(J)
      READ (2,1) DUMMY
      READ (2,1) DUMMY
                 READ (2,*) ASPECT(J)
      READ (2,1) DUMMY
      READ (2,1) DUMMY
            READ (2,*) POROS(J)
10    READ (2,1) DUMMY
      REWIND 2
      RETURN
      END
*************************************************************************
* SUBROUTINE NOPAD * GETS RID OF BLANKS IN NAMES  **********
*************************************************************************
      SUBROUTINE NOPAD(NAME)
      CHARACTER*70 OLD,NEW,NAME
      CHARACTER*1 O(70),N(70)
      INTEGER I,J
      EQUIVALENCE (OLD,O(1))
      EQUIVALENCE (NEW,N(1))
      OLD=NAME
      NEW='
     *     '
      J=0
      DO 10 I=1,70
      N(I)=' '
      IF (O(I) .NE.' ') THEN
      J=J+1
      N(J)=O(I)
      END IF
10    CONTINUE
      NAME=NEW
      RETURN
      END

*************************************************************************
* SUBROUTINE ZPTRAN * DO THE IMPEDANCE TRANSLATION THEOREM ****
* *DO ALSO THE PRESSURE TRANSLATION THEOREM ****
* THIS VERSION IS FOR THE HEAT EXCHANGERS AND OPEN TUBE SECTIONS.
*************************************************************************
      SUBROUTINE ZPTRAN(ZINT,k,r,d,ar,Z,ZMD,P1,P1MD)
      double COMPLEX Z,ZMD,ZINT,SN,CS,CT,P1,P1MD,FAC,ar
      double complex k,arg,i
```

```
      double precision r,d
      integer m,n
            character*70 geometry,fluid,solid,config
            common /switchs/ geometry,fluid,solid,config
      FAC = ZINT / Z
            cs = cdcos(ar)
            sn = cdsin(ar)
      CT = CS / SN
      ZMD = ZINT * ( CT - (0.D0,1.D0)*FAC ) /
     * ( FAC*CT - (0.D0,1.D0) )
      P1MD = P1 * ( CS - (0.D0,1.D0) * FAC * SN )
!
      RETURN
      END
*************************************************************************
* SUBROUTINE DERIVS * COMPUTE THE DERIVATIVES DZ/DZ AND DP/DZ FOR
* THE RUNGE-KUTTA WORK.
*************************************************************************
      SUBROUTINE DERIVS(ZETA,ZINT,ALPRIM,P,Z,DPDZ,DZDZ)
      COMPLEX*16 ZETA,ZINT,ALPRIM,P,Z,DPDZ,DZDZ,I,FAC,FAC2
      I = (0.D0,1.D0)
      FAC = Z/ZINT
      FAC = (1.D0 - FAC*FAC)
      FAC2 = I * ZETA * ZINT
      DZDZ = FAC2 * FAC + 2.D0 * ALPRIM * Z
      DPDZ = FAC2 * P / Z
      RETURN
      END
*************************************************************************
* SUB STAKPM * GET THE MANY PARAMETERS WHICH ARE TEMPERATURE
DEPENDENT
* IN THE STAK. USED FOR RUNGE KUTTA INTEGRATION.
*************************************************************************
      SUBROUTINE STAKPM(ETYPE,W,POROUS,PAMB,T,T0Z,DENS,RATIO,ASPECT,
     *            FLAM,FLAMT,ZETA,ALPRIM,ZINT,zcoor)
      CHARACTER*70 ETYPE
      double precision POROUS,PAMB,T,T0Z,DENS,RATIO,ASPECT,zcoor
      double complex FLAM,FLAMT,ZETA,ALPRIM,ZINT,W,LAMBDA,LAMBDT
* SUPPORTING ROLE VARIABLES.
      double precision SSPEED,VISC,CP,NPR,GAMMA,KGAS,KSOLID
            character*70 geometry,fluid,solid,config
            common /switchs/ geometry,fluid,solid,config
      COMMON /PHYCON/ GAMMA,NPR,CP
* BEGIN
      CALL VDCHE(T,PAMB,VISC,DENS,SSPEED,KGAS,KSOLID)
      CALL GETLAM(DENS,VISC,W,RATIO,LAMBDA)
            LAMBDAT = DSQRT(NPR) * LAMBDA
            IF (ETYPE.EQ.'SLIT') THEN
```

```
      CALL FSLIT(LAMBDA,FLAM)
          CALL FSLIT(LAMBDT,FLAMT)
        ELSE
          print*,aspect,lambda,flam
          call frect(aspect,lambda,flam)
          call frect(aspect,lambdat,flamt)
          print*,flam,flamt
        END IF
    CALL WNHEX(FLAMT,FLAM,W,SSPEED,ZETA)
    ZINT = DENS*W / (POROUS * FLAM * ZETA)
    ALPRIM = T0Z * (FLAMT / FLAM - 1.D0) / (2.D0 * T * (1.D0 - NPR))
    RETURN
    END
*****************************************************************************
*  SUBROUTINE ZFREE ** IMPEDANCE OF an open tube TERMINATION.
*****************************************************************************
        subroutine zfree(ratio,wavnum,sspeed,dens, Z)
        implicit double precision (a-h,o-z)
        double complex ka,Z,wavnum
        KA = wavnum * RATIO
        Z = SSPEED*DENS*KA*(KA/4.D0 - (0.0D0,0.6D0))
        return
        end
*****************************************************************************
*  SUBROUTINE ZFLANGED ** IMPEDANCE OF an open tube TERMINATION with flang.
*****************************************************************************
        subroutine zflanged(ratio,wavnum,sspeed,dens, Z)
        implicit double precision (a-h,o-z)
        double complex ka,Z,wavnum
        KA = wavnum * RATIO
        Z = SSPEED*DENS*KA*(KA/2.D0 - (0.0D0,0.8488D0))
        return
        end
*****************************************************************************
*  SUBROUTINE Zinfin ** IMPEDANCE OF an infinite tube.
*****************************************************************************
        subroutine zinfin(dens,w,flam,wavnum, Z)
        implicit double precision (a-h,o-z)
        double complex Z,wavnum,W,flam
        Z = DENS * W / (FLAM * wavnum)
        return
        end
*****************************************************************************
*  SUBROUTINE ZRIGID ** IMPEDANCE OF A RIGID TERMINATION.
*****************************************************************************
    SUBROUTINE ZRIGID(DENS,SSPEED,VISC,W,Z)
    double precision DENS,SSPEED,VISC,NPR,GAMMA,CP
    double COMPLEX Z,W,FAC
```

```
      COMMON /PHYCON/ GAMMA,NPR,CP
      FAC = CDSQRT( DENS*SSPEED**2 / (W*VISC) )
      Z = (1.D0,1.D0)*DENS*SSPEED*FAC*DSQRT(NPR)/(DSQRT(2.D0)*(GAMMA -
     *           1.0D0))
      RETURN
      END
*************************************************************************
* SUBROUTINE WNTUBE  WAVENUMBERS FOR WAVES IN THE OPEN TUBE PARTS.
*************************************************************************
      SUBROUTINE WNTUBE(LAMBDA , W , SSPEED , K)
      double precision SSPEED,GAMMA,NPR,FAC1,CP
      double COMPLEX K,W,lambda
      COMMON /PHYCON/ GAMMA,NPR,CP
      FAC1 = ( 1.D0 + (GAMMA - 1.D0) / DSQRT(NPR) ) /  DSQRT(2.D0)
      K = W/SSPEED * ( 1.D0 + (1.D0 , 1.D0) * FAC1 / LAMBDA  )
      RETURN
      END
*************************************************************************
* SUBROUTINE FTUBE * COMPUTES F(LAMBDA) FOR THE RESONANT TUBE.
*************************************************************************
      SUBROUTINE FTUBE(LAMBDA,FLAM)
      double COMPLEX LAMBDA,FLAM
      FLAM = 1.D0 - (1.D0,1.D0) * DSQRT(2.0D0) / LAMBDA
      RETURN
      END
*************************************************************************
* SUBROUTINE WNHEX  WAVENUMBERS FOR WAVES IN THE HEAT EXCHANGERS.
*************************************************************************
      SUBROUTINE WNHEX(FLAMT , FLAM , W , SSPEED , K)
      double precision SSPEED,GAMMA,NPR,CP
      double COMPLEX FLAMT,FLAM,K,W
      COMMON /PHYCON/ GAMMA,NPR,CP
      K = W/SSPEED * CDSQRT( (GAMMA - (GAMMA - 1.D0)*FLAMT) / FLAM )
      RETURN
      END
*************************************************************************
* SUBROUTINE VDCHE ** VISCOSITY, DENSITY, AND SOUND SPEED OF the
* working fluid,
* AS A FUNCTION OF TEMPERATURE AND AMBIENT PRESSURE. ALSO THE THERMAL
* CONDUCTIVITY of the gas and solid material, currently polyimad.
*************************************************************************
      SUBROUTINE VDCHE(TABS,PAMB,VISC,DENS,SSPEED,KGAS,KSOLID)
      implicit double precision(a-h,o-z)
      double precision TABS,PAMB,VISC,DENS,SSPEED
      double precision GAMMA,NPR,CP,KGAS,KSOLID
      double precision k0He,k0Ar
      character*70 geometry,fluid,solid,config
      parameter(Ts=110.4d0,Ta=245.4d0,Tb=27.6d0,
```

```
*     T0=300.d0,Texp=223.8306d0)
      common /switchs/ geometry,fluid,solid,config
      COMMON /PHYCON/ GAMMA,NPR,CP
      Runiv = 8.3143d0
! CASE OF HELIUM FIRST.
      if (fluid.eq.'HELIUM') then
       um = 4.d-3 ! molecular weight in Kgrams / mole.
       NPR = 2.D0 / 3.D0
       GAMMA = 5.D0 / 3.D0
       CP = 2.5D0 * Runiv / um
       DENS = PAMB * um / ( TABS * Runiv )
* MY EXPRESSION FOR VISCOSITY.
       VISC = 1.887D-5 * (TABS / 273.15D0)**0.6567D0
* MY EXPRESSION FOR THERMAL CONDUCTIVITY:  NPR = CONSTANT.
       KGAS = VISC * CP / NPR
* SWIFT'S EXPRESSION FOR VISCOSITY.
*    VISC = 5.131D-7 * TABS**.6441D0
* SWIFT'S EXPRESSION FOR THERMAL CONDUCTIVITY. NPR NOT CONSTANT.
*     KGAS = 0.0044 * TABS**.6441D0
*     NPR = VISC * CP / KGAS
       SSPEED = 972.8D0 * DSQRT(TABS / 273.15D0)
! CASE OF AIR NEXT.
      else if (fluid.eq.'AIR') then
       um = 29.d-3 ! molecular weight in Kgrams / mole.
       CP = 3.5D0 * Runiv / um
       GAMMA = 7.D0 / 5.D0
       DENS = PAMB * um / ( TABS * Runiv )
       SSPEED =  DSQRT(GAMMA*Runiv*TABS / um)
* Pierce's EXPRESSION FOR VISCOSITY and thermal conductivity.
       p=(Tabs/T0)**1.5d0
       VISC = 1.846d-5*p*(T0+Ts)/(Tabs+Ts)
       KGAS = 2.624d-2*p*(T0+Texp)/(Tabs+Ta*dexp(-Tb/Tabs))
       NPR = VISC * CP / KGAS
! CASE OF HELIUM ARGON MIX NEXT. xhe=mole fraction helium.
      else if (fluid.eq.'HE60AR40') then
       gamma=5.d0/3.d0
       umHe = 4.d-3
       umAr = 40.d-3
       xhe = 0.6d0 ! Mole Fraction of helium.
       xAr = 1.d0 - xhe ! Mole Fraction of Argon.
       aveum = xhe*umHe + xAr*umAr
       sspeed=dsqrt(gamma*Runiv*Tabs / aveum)
       dens=Pamb*aveum/(Runiv*Tabs)
       CP = 2.5d0*Runiv/aveum
! Use DELTAE transport properties.
       k0He=0.0025672d0*(Tabs**0.716d0)
       amuHe=0.412d-6*(Tabs**0.68014d0)
       k0Ar=(1.39d-4*(Tabs**0.852d0)-1.5d-8*(Tabs-300.d0)*
```

```
     *   (Tabs-300.d0))*(1.d0 + 2.d-8*Pamb)
           amuAr=(1.77d-7*Tabs**0.852-25.d-12*(Tabs-300.d0)*
     *   (Tabs-300.d0))*(1.d0+2.d-8*Pamb)
           Kgas=xAr*k0Ar+xHe*k0He-(k0Ar+K0He)*xAr*xHe**1.5d0
           VISC=xAr*amuAr+xHe*amuHe+0.2d0*(amuAr+amuHe)*xAr*xHe
           NPR = VISC * CP / KGAS
          end if ! determining fluid type.
* DETERMINE WHAT SOLID IS USED in the stack and get its thermal conduct.
! Units:  watts / (meter Kelvin).
          if (solid.eq.'KAPTON') then
           KSOLID=0.2d0*(1.d0 - dexp(-Tabs/100.d0))
          else if (solid.eq.'POLYIMID') then
           KSOLID=0.35d0*(1.d0 - dexp(-Tabs/100.d0)) ! T dep from kapton
          else if (solid.eq.'STAINLESS') then
           KSOLID=(3.64187d0+0.00267962*(Tabs-273.d0)+
     *      4.49327d-7*(Tabs-273.d0)**2)*4.186d0
          else if (solid.eq.'COPPER') then
           KSOLID=398.d0 - 0.0567*(Tabs-300.d0)
          else if (solid.eq.'CELCOR') then
           KSOLID=0.92d0 ! From DeLuca and Campdell, pg 304.
          end if
          RETURN
          END
*************************************************************************
* SUBROUTINE FSLIT *** COMPUTES F(LAMBDA) FOR PARALLEL SLITS.
*************************************************************************
      SUBROUTINE FSLIT(LAMBDA , FLAM)
      double COMPLEX LAMBDA,FLAM,SQRMI,ARGUM,CTANH,AR
      SQRMI = (1.0D0 , -1.0D0) / DSQRT( 2.0D0 )
      AR = SQRMI * LAMBDA
      ARGUM = CDEXP(-AR)
      AR = AR / 2.D0
      CTANH = ( 1.D0 - ARGUM ) / ( 1.D0 + ARGUM )
      FLAM = 1.0D0 - CTANH / AR
      RETURN
      END
*************************************************************************
* SUBROUTINE FCYL **** COMPUTES F(LAMBDA) FOR CYLINDRICAL PORES.
*************************************************************************
      SUBROUTINE FCYL(LAMBDA , FLAM)
      double COMPLEX FLAM,SQRI,CBS(2),J0,J1,ARGUM,LAMBDA
      INTEGER N
      N=2
      SQRI = (1.0D0 , 1.0D0) / DSQRT( 2.0D0 )
      ARGUM = SQRI * LAMBDA
      CALL CBESJ(argum , N, CBS,IERR)
      if (ierr.ne.0) write(*,*) 'fcyl cbesj ierr=',ierr
      J0 = CBS(1)
```

```
      J1 = CBS(2)
      FLAM = 1.0D0 - ( 2.0D0 * J1 ) / ( ARGUM * J0 )
      RETURN
      END
*************************************************************************
* SUBROUTINE FRECT **** COMPUTES F(LAMBDA) FOR RECTANGULAR PORES.
*************************************************************************
      SUBROUTINE FRECT(ASPECT , LAMBDA , FLAM)
      double precision PI,ASPECT,ASPSQ
      double precision XN,XM,FAC
      INTEGER M,N,SUMMAX
      double COMPLEX SUM,FLAM,YMN,LAMBDA,FAC1
      PI = 4.0D0 * DATAN(1.0D0)
      FAC1 = PI * PI / ( LAMBDA * ( 1.0D0 + ASPECT ) )**2
           FAC = 64.0D0 / PI**4
           ASPSQ = ASPECT * ASPECT
      SUMMAX = 25  ! must be an odd number!
      SUM = (0.0D0 , 0.0D0)
      DO 30 M=SUMMAX,1,-2
      XM = DFLOAT(M)
      XM = XM*XM
      DO 40 N=SUMMAX,1,-2
      XN = DFLOAT(N)
      XN = XN*XN
      YMN = 1.D0 + (0.D0,1.D0) * FAC1 * ( ASPSQ * XM + XN )
40    SUM = SUM + 1.0D0 / (  XM * XN * YMN )
30    CONTINUE
      FLAM = SUM * FAC
      RETURN
      END
*************************************************************************
* SUBROUTINE QUAFAC COMPUTES THE QUALITY FACTOR AND RESONANT FREQU.
*************************************************************************
      SUBROUTINE QUAFAC(AMP,FREQ,Q,RESFRE,numfreq)
      double precision Q,RESFRE,AMP(2000),FREQ(2000)
           double precision MAXAMP,FREHAF,AMPHAF
      double precision AMP2(2000),FREQ2(2000)
*       real*8 XC(3),BL(3),BU(3),XSCALE(3)
*      REAL*8 XGUESS(3),FVALUE,FSCALE(3),RPARAM(7)
      INTEGER J,JRES,JHALF,NUMDAT,JCOUNT,N,ISTART,IBTYPE,IPARAM(7)
           integer numfreq
      COMMON /QCALC/ AMP2,FREQ2,NUMDAT
      EXTERNAL FUNCT1
      N = 3
      IPARAM(1) = 0
      IBTYPE=0
      ISTART=0
      MAXAMP = 0.D0
```

```
      JCOUNT = 0
      DO 10 J=1,numfreq
      IF (AMP(J) .GT. MAXAMP) THEN
      MAXAMP=AMP(J)
      RESFRE = FREQ(J)
      JRES = J
      END IF
10    CONTINUE
      AMPHAF=MAXAMP/DSQRT(2.D0)
      DO 20 J=1,numfreq
      IF (AMP(J) .GT. AMPHAF) THEN
      FREHAF = (FREQ(J)+FREQ(J-1))/2.D0
      Q = 0.5D0 * RESFRE / (RESFRE - FREHAF)
      JHALF = J
      GOTO 30
      END IF
20    CONTINUE
30    DO 40 J=JHALF-1,JRES + (JRES-JHALF) + 1
      JCOUNT = JCOUNT + 1
      AMP2(JCOUNT) = AMP(J)
40    FREQ2(JCOUNT) = FREQ(J)
      NUMDAT = JCOUNT
            return
*     XGUESS(1) = MAXAMP
*     XGUESS(2) = RESFRE
*     XGUESS(3) = Q
*     DO 50 J=1,3
*     XSCALE(J)=1.D0
*     FSCALE(J)=1.D0
*     BL(J) = XGUESS(J) * .75D0
*50   BU(J) = XGUESS(J) * 1.75D0
*     CALL DBCONF(FUNCT1, N , XGUESS , IBTYPE , BL , BU ,
*    *          XSCALE , FSCALE , IPARAM , RPARAM , XC , FVALUE)
*     MAXAMP = XC(1)
*     RESFRE = XC(2)
*     Q = XC(3)
*     RETURN
      END
*******************************************************************************
****************************** SUBROUTINE FUNCT1 FOR IMSL OPTIMIZATION
*******************************************************************************
      SUBROUTINE FUNCT1(N , XC , RMSERR)
      double precision AMP(2000),FREQ(2000),XC(3)
           double precision RMSERR,MAXSQ,F0,Q,F
      INTEGER NUMDAT,J
      COMMON /QCALC/ AMP,FREQ,NUMDAT
      MAXSQ = XC(1)*XC(1)
      F0 = XC(2)
```

```
      Q = XC(3)
      RMSERR = 0.D0
      DO 10 J=1,NUMDAT
      F = FREQ(J)
      RMSERR = ( MAXSQ / (1.D0 + (2.D0*Q*(F-F0)/F0)**2 )
     *        - AMP(J)*AMP(J))**2 + RMSERR
10    CONTINUE
      RETURN
      END
************************************************************************
*************************** SUBROUTINE GETLAM ***********************
*************************** LAMBDA **********************************
      SUBROUTINE GETLAM(DENS , VISC , W , R , LAMBDA)
      double precision DENS,VISC,R
      double COMPLEX LAMBDA,W
      LAMBDA = R * CDSQRT( DENS * W / VISC )
      RETURN
      END
************************************************************************
* SUBROUTINE CHANGE.  USED TO INSERT A NUMBER INTO A LINE OF A
* SEQUENTIAL FILE.
************************************************************************
      SUBROUTINE CHANGE(numvar,FNUM,LNUM,TO1,to2,to3)
      INTEGER FNUM,LNUM,J,numvar
      CHARACTER*80 LINE
1     FORMAT(A80)
      double precision TO1,to2,to3
      OPEN(73,FILE='scratch')
      REWIND(73)
      REWIND(FNUM)
      DO 10 J=1,5000
      READ(FNUM,1,END=20) LINE
      IF (J.NE.LNUM) THEN
      WRITE(73,1) LINE
      ELSE
           if (numvar.eq.1) then
      WRITE(73,*) TO1
           else if (numvar.eq.2) then
      WRITE(73,*) TO1,to2
           else if (numvar.eq.3) then
      WRITE(73,*) TO1,to2,to3
           end if
      END IF
10    CONTINUE
20    ENDFILE 73
           close(73)
! Duplicating scratch to fnum.
*     REWIND(3)
```

```
*     REWIND(FNUM)
*     DO 30 J=1,5000
*     READ(3,1,END=40) LINE
*30    WRITE(FNUM,1) LINE
*40    ENDFILE FNUM
*     REWIND(3)
*     REWIND(FNUM)
*      close(3)
      RETURN
      END


*************************************************************************
* Subroutine scan.
* Does scans of variables such as frequency, plate spacing, etc
*************************************************************************
      subroutine scan
      implicit double precision (a-h,o-z)
      real xnewt(2),fvec(2)
      real xnewtpass(2)
      real d,din,dcm,dmil,dmm,r,rin,rmil,rcm,rmm
      real t,tin,tcm,tmil,tmm,out1,out2,out3
      logical check
      character*1 answer
      character*70 config,configold,geometry,fluid,solid
      double precision delem(100),heatload(100),Trgh(100),
     *  Tleft(100),ratio(100),aspect(100),poros(100)
      double precision ratios(100)
      PARAMETER (N=5000)
      REAL*8 T0Z(n),TAVE(N),W2(N),zcoor(n)
      COMPLEX*16 P1(N),Z(N)
      integer inod,ir(100),il(100)
      double precision trghs(100),tlefts(100)
      double precision KSOLID,KGAS
      INTEGER NUMLAY(100),NUMLAYS(100),NUMSEC
      CHARACTER*70 SECTYP(100),ETYPE(100),TERMINR,TERMINL
      character*50 filename,filenew
      character*70 command
      double precision NPR
      common /xnewtpass/ xnewtpass
      common /switchs/ geometry,fluid,solid,config
      COMMON /OUTPUT/ P1,Z,zcoor,W2,TAVE,T0Z
      common /account/ numtot
      common /shstack/ phaseangle,Wext
      COMMON /VARS1/ SECTYP,ETYPE,TERMINR,TERMINL
      COMMON /VARS2/ NUMLAY,NUMSEC,NUMFRE
     * ASPECT,POROS,PAMB,FREMIN,FREMAX,p1not
      COMMON /PHYCON/ GAMMA,NPR,CP
```

```
      pi=4.d0*datan(1.d0)
      twopi=2.d0*pi
1     format(a1)
      nnewt=2 ! Two variables in the newton raphson itteration.
      call TAE(f0est,dT,qual) ! Estimate for frequency,dT,qual.
      configold=config ! Configuration storage.   Subroutine minimize
*Parameters needed for amoeba.
      double precision d1,dnumsec,facmin,facmax
      double precision pamb,p1not
      real p(3,2),y(3),ftol,x(2)
      integer ndim,mp,np
      double precision delem(100),heatload(100),trgh(100),Tleft(100)
      *  ,ratio(100),aspect(100),poros(100)
      INTEGER NUMLAY(100),NUMSEC
      external funk
      COMMON /VARS2/ NUMLAY,NUMSEC
      COMMON /VARS3/ DELEM,HEATLOAD,TRGH,TLEFT,RATIO,
      *  ASPECT,POROS,PAMB,p1not
*Variables needed by amoeba.
      d1=delem(1)
      dnumsec=delem(numsec)
      ndim=2
      mp=ndim+1
      np=ndim ! number of dimensions in the amoeba.
      ftol=1.e-3
*Begin setting initial values for amoeba.
      facmin=0.95d0
      facmax=1.05d0
*ndim is the number of variables to be evaluated.
      do 20 j=1,ndim+1
      fac=facmin + (facmax-facmin)*dfloat(j-1)/dfloat(ndim+1)
      x(1)=real(d1*fac)
      p(j,1)=x(1)
      x(2)=real(dnumsec*fac)
      p(j,2)=x(2)
      y(j)=funk(x)
20    write(*,*) 'y(',j,')=',y(j)
      call amoeba(p,y,mp,np,ndim,ftol,funk,iter)
      delem(1)=real(p(1,1))
      delem(numsec)=real(p(1,2))
      return
      end

**************************************************************************
**************************************************************************
* VERSION 3.0 FOR HELIUM BY PAT ARNOTT, 16 FEB 91  ********************
* THIS VERSION USES RUNGE KUTTA SOLUTION FOR THE DE INSIDE OF THE STAK.
```

```
* TRANSLATION THEOREMS ARE USED IN OPEN TUBE AND HEAT EXCHANGERS.
* W THE RADIAN FREQUENCY IS ASSUMED COMPLEX EVERYWHERE.
* AUGMENTED NOV 93 FOR USE ON RADIAL WAVE PRIME MOVERS, OR ONEDIM
* PRIME MOVERS, AS DETERMINED BY THE VARIABLE GEOMETRY IN THE COMMON
* SWITCH.
*************************************************************************
      SUBROUTINE TAE(f0est,dT,qual)
      implicit double precision (a-h,o-z)
      double complex Zhole,p1dumb,p1temp,Ztemp,p1trans,ztrans
      double complex Zinthole,zetahole
      double COMPLEX W,LAMBDA,LAMBDT
      double complex Zsaimatch
* GENERIC VARIABLES SUCH AS PHYSICAL PROPERTIES OF THE GAS.
      double precision SSPEED,VISC,CP,NPR,GAMMA,KGAS,KSOLID
* Z DEPENDENT ARRAYS....
      PARAMETER (N=5000)
      double precision TAVE(N),T0Z(N),ZCOOR(N),W2(N)
      double COMPLEX FLAM,FLAMT,Z(N),P1(N)
* VARIABLES WHICH DEFINE THE TAE.
      CHARACTER*70 SECTYP(100),ETYPE(100),TERMINR,TERMINL
      INTEGER NUMLAY(100),NUMSEC
      double precision DELEM(100),TRGH(100),TLEFT(100),RATIO(100)
     * ,ASPECT(100),POROS(100),heatload(100)
     * ,PAMB
* DEFINE SOME GLOBAL VARIABLES.
      double precision PI,TWOPI
      INTEGER nnewt
      real xnewt(2),fvec(2)
      real xnewtpass(2)
* EXTRA VARIABLES NECESSARY FOR RUNGE KUTTA EVALUATION OF THE PROBLEM.
      double COMPLEX K1,K2,K3,K4,M1,M2,M3,M4,DPDZ,DZDZ
      double COMPLEX ZETA,ALPRIM,ZINT,AR,SN,CS
      double precision TNS,TNSM1
      INTEGER I,J,JUP,JLOW,NUMTOT
      character*70 geometry,config,fluid,solid
      common /switchs/ geometry,fluid,solid,config
      common /xnewtpass/ xnewtpass
      COMMON /PHYCON/ GAMMA,NPR,CP
      common /shstack/ phaseangle,Wext
      COMMON /VARS1/ SECTYP,ETYPE,TERMINR,TERMINL
      COMMON /VARS2/ NUMLAY,NUMSEC
      COMMON /VARS3/ DELEM,HEATLOAD,TRGH,TLEFT,RATIO,
     *  ASPECT,POROS,PAMB,p1not
      COMMON /OUTPUT/ P1,Z,zcoor,W2,TAVE,T0Z
      common /account/ numtot
* ESTABLISH SOME OFTEN USED CONSTANTS.
* Check to see if tae has been initialized, and if so jump to guess.
11111  format(a1)
```

```
      if ((config.eq.'wallend').or.(config.eq.'center').or.
     * (config.eq.'quality')) goto 1999
      PI = 4.D0 * DATAN(1.D0)
      TWOPI = 2.D0 * PI
* Determine which is the largest R element to compute ARES with.
      Rmax=0.d0
      do 600 i=1,numsec
600   if (ratio(i).gt.Rmax) Rmax=ratio(i)
      Ares=pi*Rmax*Rmax
* Initialize the transport and response functions of the gas.
* Determine which end, near the wall, or near the center, is to have
* its temperature adjusted to reach onset.
* Or perhaps one just wants the resonant frequency and Q.
      if (Trgh(1).eq.0.d0) then
       config='center'
       Tambient=Trgh(numsec)
       dT = -90.d0 ! Onset temperature guess ABOVE ambient!
! Set the temperature of all other elements to ambient to start off.
       do 94 i=1,numsec-1
       Trgh(i)=Tambient
94     Tleft(i)=Tambient
       write(*,*) 'ONSET TEMP. GUESS =',dT+Tambient-273.d0,' C'
      else if (Trgh(numsec).eq.0.d0) then
       config='wallend'
       Tambient = Trgh(1)
       dT = 150.d0 ! Onset temperature guess ABOVE ambient!
! Set the temperature of all other elements to the ambient to start off.
       do 93 i=numsec,2,-1
       Trgh(i)=Tambient
93     Tleft(i)=Tambient
       write(*,*) 'ONSET TEMP. GUESS =',dT+Tambient-273.d0,' C'
      else
       config='quality'
       write(*,*) 'Press 1 to guess Q, 2 for default Q = 100   : '
       read(*,*) ians
       if (ians.eq.1) then
        write(*,*) 'Enter quality factor guess, eg. 100    : '
        read(*,*) qual
        else
        qual=100.d0
       end if
       write(*,*) 'Quality factor GUESS =',qual
       Tambient=0.d0
       do 97 i=1,numsec
97     Tambient=Tambient + Trgh(i) + Tleft(i)
       Tambient=Tambient/(2.d0*dfloat(numsec))
      end if
      dT=dT + Tambient ! Set the initial guess to Kelvin.
```

```
        dTold=dT
        qualold = qual
1999    write(*,*)
        qual=qualold
        dT=dTold
        if (config.eq.'quality') then
        write(*,*) 'qual =',qual
        else
        end if
        CALL VDCHE(Tambient,PAMB,VISC,DENS,SSPEED,KGAS,KSOLID)
* ESTIMATE THE RESONANT FREQUENCY FROM THE LENGTH AND SOUND SPEED
DIST.
        OPL = 0.0
* Estimate f0est from impedance translation theorem.
        DO 1 i=1,NUMSEC
        Taverage = (Tleft(i)+Trgh(i))/2.d0
        CALL VDCHE(Taverage,PAMB,VISC,DENS,SSPEED,KGAS,KSOLID)
1       OPL = OPL + delem(i)/sspeed
* Determine the mode.... for setting the initial frequency guess.
        if (((TERMINL.eq.'NODE').or.(TERMINL.eq.'RIGID')).and.
     * (TERMINR.eq.'RIGID')) then
         F0EST = 1.d0 / (2.D0 * opl)
        else if (((TERMINL.eq.'NODE').or.(TERMINL.eq.'RIGID')).and.
     * (TERMINR.eq.'FREE')) then
         F0EST = 1.d0 / (4.D0 * opl)
        else if ((TERMINL.eq.'FREE').and.
     * (TERMINR.eq.'FREE')) then
         F0EST = 1.d0 / (2.D0 * opl)
        else if ((TERMINL.eq.'FREE').and.
     * (TERMINR.eq.'RIGID')) then
         F0EST = 1.d0 / (4.D0 * opl)
        else
         F0EST = 1.d0 / (2.D0 * opl) !DEFAULT.
        end if
        write(*,*) 'Resonant frequency guess =',f0est,' Hz'
        RETURN
        ENTRY funcv(nnewt,xnewt,fvec)
        w=twopi*real(xnewt(1))
        if (config.eq.'quality') then ! Aiming at determining Q.
         w=w - (0.d0,1.d0)*xnewt(2)
        else ! Aiming at onset....
         Thot = real(xnewt(2))
         if (config.eq.'wallend') then
          do 19 i=numsec,1,-1
          if (sectyp(i).eq.'STACK') goto 1923
          Trgh(i)=Thot
19       Tleft(i)=Thot
1923     Trgh(i)=Thot
```

```
          else if (config.eq.'center') then
           do 29 i=1,numsec
           if (sectyp(i).eq.'STACK') goto 1924
           Trgh(i)=Thot
29        Tleft(i)=Thot
1924       Tleft(i)=Thot
           end if
          end if ! if config.eq.'quality'
* GET THE SPECIFIC ACOUSTIC IMPEDANCE AND PRESSURE AT ALL POINTS.
* START AT THE RIGHT AND MOVE TO THE LEFT.
          NUMTOT = 0
          zcoormax=0.d0
          do 10 i=1,NUMSEC
          zcoormax=zcoormax + delem(i)
10      NUMTOT = NUMTOT + NUMLAY(i)
          NUMTOT = NUMTOT + 1
          CALL VDCHE(TRGH(NUMSEC),PAMB,VISC,DENS,SSPEED,KGAS,KSOLID)
          CALL GETLAM(DENS,VISC,W,RATIO(numsec),LAMBDA)
          LAMBDT = DSQRT(NPR) * LAMBDA
          CALL FTUBE(LAMBDA,FLAM)
          CALL FTUBE(LAMBDT,FLAMT)
          CALL WNTUBE(LAMBDA,W,SSPEED,ZETA)
          if (TERMINR.eq.'RIGID') then
           CALL ZRIGID(DENS,SSPEED,VISC,W,Z(NUMTOT))
          else if (TERMINR.eq.'FREE') then
           call zfree(ratio(numsec),ZETA,sspeed,dens, Z(numtot))
          else if (TERMINR.eq.'INFIN') then
           call zinfin(dens,w,flam,ZETA, Z(numtot))
          else
           write(*,*) 'EDIT params file since TERMINR is set wrong'
          end if
          P1(NUMTOT) = dcmplx(p1not,0.d0) ! Pressure at the right wall.
          TAVE(NUMTOT) =  TRGH(NUMSEC)
          T0Z(NUMTOT) = 0.D0
          zcoor(numtot)=zcoormax ! Can be z or r, as plane or radial wave.
          W2(numtot)=cdabs(p1(numtot))**2*
     *  real(Z(numtot))/cdabs(Z(numtot))**2
          W2(numtot)=W2(numtot)*Ares/2.d0
! Initialize the temporary variables.
          Ztrans=Z(numtot)
          p1trans=P1(numtot)
* APPLY THE IMPEDANCE AND PRESSURE TRANSLATION THEOREMS EVERYWHERE.
* WORK FROM RIGHT TO LEFT.
     DO 40 I=NUMSEC,1,-1.
     JUP = 0
     DO 50 J=I+1,NUMSEC
50    JUP = JUP + NUMLAY(J)
     JUP = NUMTOT - JUP - 1
```

```
      JLOW = JUP - NUMLAY(I) + 1
      DSUB = DELEM(i) / dfloat(NUMLAY(i))
* IMPEDANCE TRANSLATE FOR all SECTIONS.
            if (sectyp(i).eq.'STACK') then ! Set up the geometrical constraints.
             rb=zcoor(jup+1) ! Right coordinate of element.
             VG=Ares*poros(i)*delem(i) ! Stack open volume.
            Tdiff=Trgh(i)-Tleft(i) !Temperature difference across stack.
            Tgrad=Tdiff/delem(i) ! One d temperature gradient across stack.
*
            else ! Section type is heat exchanger or open tube.
            CALL VDCHE(TRGH(i),PAMB,VISC,DENS,SSPEED,KGAS,KSOLID)
            TAVE(JUP) = TRGH(i)
            T0Z(JUP) = 0.0D0
            CALL GETLAM(DENS,VISC,W,RATIO(i),LAMBDA)
            LAMBDT = DSQRT(NPR) * LAMBDA
            IF (SECTYP(i).EQ.'OPENTU')  THEN
              CALL FTUBE(LAMBDA,FLAM)
              CALL FTUBE(LAMBDT,FLAMT)
              CALL WNTUBE(LAMBDA,W,SSPEED,ZETA)
* OTHERWISE, THE TUBE SECTION IS A HEAT EXCHANGER.  FIND ITS GEOMETRY.
            ELSE IF (SECTYP(i).EQ.'HEXCH') THEN
              CALL FSLIT(LAMBDA,FLAM)
              CALL FSLIT(LAMBDT,FLAMT)
              CALL WNHEX(FLAMT,FLAM,W,SSPEED,ZETA)
         END IF ! if sectyp.eq.opentu.
             ZINT = dens * W / ( ZETA * FLAM * POROS(i))
            end if ! if sectyp eq 'stack'.
            ztrans=Z(jup+1)
            p1trans=P1(jup+1)
      DO 70 J=JUP,JLOW,-1
            zcoor(j)=zcoor(j+1)-dsub
            if (sectyp(i).eq.'STACK') then ! get the temp at stack points.
             zmid=zcoor(j) + dsub/2.d0
! Describe the temperature gradient, and temperature at the right, middle
! and left end of element of length dsub(j) for the runge kutta
! integrations.
             Tnsm1=Trgh(i)-Tgrad*(rb-zcoor(j+1)) ! Right end temperature.
             Tns=Trgh(i)-Tgrad*(rb-zcoor(j)) ! Left end temperature.
             Tmid=Trgh(i)-Tgrad*(rb-zmid) ! Mid point temperature.
             Tgnsm1=Tgrad ! Right end temperature gradient.
             Tgns=Tgrad ! left end temperature gradient.
             Tgmid=Tgrad ! Midpoint temperature gradient.
* START THE RUNGE-KUTTA
             Tave(j)=Tns
             T0z(j) = Tgns
       CALL STAKPM(ETYPE(I),W,POROS(i),PAMB,TNSM1,Tgnsm1,DENS,
     2 RATIO(I),ASPECT(I),FLAM,FLAMT,ZETA,ALPRIM,ZINT,zcoor(j+1))
       CALL DERIVS(ZETA,ZINT,ALPRIM,P1(J+1),Z(J+1),DPDZ,DZDZ)
```

```
      K1 = -DSUB*DPDZ
      M1 = -DSUB*DZDZ
      P1(J) = P1(J+1) + K1/2.D0
      Z(J)   = Z(J+1) + M1/2.D0
       CALL STAKPM(ETYPE(I),W,POROS(i),PAMB,Tmid,Tgmid,DENS,
     2 RATIO(I),ASPECT(I),FLAM,FLAMT,ZETA,ALPRIM,ZINT,zmid)
      CALL DERIVS(ZETA,ZINT,ALPRIM,P1(J),Z(J),DPDZ,DZDZ)
      K2 = -DSUB*DPDZ
      M2 = -DSUB*DZDZ
      P1(J) = P1(J+1) + K2/2.D0
      Z(J)   = Z(J+1) + M2/2.D0
      CALL DERIVS(ZETA,ZINT,ALPRIM,P1(J),Z(J),DPDZ,DZDZ)
      K3 = -DSUB*DPDZ
      M3 = -DSUB*DZDZ
      P1(J) = P1(J+1) + K3
      Z(J)   = Z(J+1) + M3
       CALL STAKPM(ETYPE(I),W,POROS(I),PAMB,TNS,Tgns,DENS,
     2 RATIO(I),ASPECT(I),FLAM,FLAMT,ZETA,ALPRIM,ZINT,zcoor(j))
      CALL DERIVS(ZETA,ZINT,ALPRIM,P1(J),Z(J),DPDZ,DZDZ)
      K4 = -DSUB*DPDZ
      M4 = -DSUB*DZDZ
      P1(J) = P1(J+1) + (K1+2.D0*K2+2.D0*K3+K4)/6.D0
      Z(J) = Z(J+1) + (M1+2.D0*M2+2.D0*M3+M4)/6.D0
            if (real(Z(j+1))*real(Z(j)).lt.0.) then ! Get phaseangle.
             zcrit = zcoor(j)-real(Z(j))*DSUB/real(Z(j+1)-Z(j))
             Zimag=Dimag(Z(j))-Dimag(Z(j+1)-Z(j))*real(Z(j)) /
     *                real(Z(j+1) - Z(j))
              Tcrit=Trgh(i)-Tgrad*(rb-zcrit)
             CALL VDCHE(Tcrit,PAMB,VISC,DENS,SSPEED,KGAS,KSOLID)
             phaseangle=datan(dens*sspeed/dabs(Zimag))
            end if
      else ! Section type is not stack, is heatexch or opentu.
            Tave(j) = Trgh(i)
            T0z(j)  = 0.d0
            ar = zeta*dsub
            call ZPTRAN(ZINT,zeta,zcoor(j+1),dsub,ar,
     *  Z(j+1),Z(j),P1(j+1),P1(j) )
            end if ! if sectyp =  stack.
            W2(j)=cdabs(p1(j))**2*real(Z(j))/cdabs(z(j))**2
            W2(j)=W2(j)*Ares/2.d0
! GET the impedance and pressure by translation for comparison purposes.
* comment out the next lines till 70.
*       Ztemp=ztrans
*       p1temp=p1trans
*       ar = dsub*zeta
*        call ZPTRAN(ZINT,zeta,zcoor(jup+1),dsub,ar,
*     *  Ztemp,Ztrans,P1temp,P1trans )
*       if (i.eq.3) then  ! Print out.
```

```
*     write(*,*) '*****************************************************'
*     write(*,*) 'p1trans',p1trans
*     write(*,*) 'p1(j)',p1(j)
*     write(*,*) 'Ztrans',Ztrans
*     write(*,*) 'Z(j)',Z(j)
*     write(*,*) 'press enter to proceed'
*     read(*,11111) du
*     end if ! if i.eq.3.
70    CONTINUE ! do j=jup downto jlow.
        if (SECTYP(i).eq.'STACK') then ! compute the external work.
         Wext=2.d0*gamma*Pamb/((real(P1(numtot))**2)*VG*real(w))
         Wext=Wext*(W2(jup+1)-W2(jlow))
        end if
40    CONTINUE ! Do i=numsec downto 1.
* The impedance is now known at the left end of the last heat exchanger
* before the tube center.  This impedance must be matched with that
* determined by the impedance at the left end of the tube.
    AR = zeta * delem(1)
        if (TERMINL.eq.'NODE') then !Velocity node at the center.
          SN = CDSIN(AR)
          CS = CDCOS(AR)
          Zsaimatch = (0.d0,-1.d0)*ZINT*CS/SN
        else
          if (TERMINL.eq.'RIGID') then
           CALL ZRIGID(DENS,SSPEED,VISC,W,Ztemp)
          else if (TERMINL.eq.'FREE') then
*         CALL ZFREE(ratio(1),zeta,sspeed,dens, Ztemp)
* Try the flanged prime mover.
           CALL ZFLANGED(ratio(1),zeta,sspeed,dens, Ztemp)
          else if (TERMINL.eq.'INFIN') then
           CALL ZINFIN(dens,w,flam,ZETA, Ztemp)
          else
           write(*,*) 'EDIT params since TERMINL is not set properly.'
          stop
          end if
          Ztemp=-Ztemp ! Sign change for particle velocity direction.
          ar=-ar ! Sign change for impedance translation in other dir.
          zero=0.d0
          dtemp=-delem(1)
          call ZPTRAN(ZINT,zeta,zero,dtemp,ar,
   *     Ztemp,Zsaimatch,P1temp,P1dumb )
        end if
        fvec(1)=real(dimag(Zsaimatch - Z(jup+1)))
        fvec(2)=real(real(Zsaimatch - Z(jup+1)))
*     kkkk=kkkk+1
*     freqtemp = real(w)/twopi
*     write(*,*) ' itter=',kkkk,' frequency',freqtemp
        xnewtpass(1)=xnewt(1)
```

```
      xnewtpass(2)=xnewt(2)
    RETURN
    END



    SUBROUTINE newt(x,n,check)
    INTEGER n,nn,NP,MAXITS
    LOGICAL check
    REAL x(n),fvec,TOLF,TOLMIN,TOLX,STPMX
    PARAMETER (NP=40,MAXITS=1500,TOLF=1.e-7,TOLMIN=1.e-6,
   *TOLX=1.e-8,STPMX=.005)
    COMMON /newtv/ fvec(NP),nn
    SAVE /newtv/
CU  USES fdjac,fmin,lnsrch,lubksb,ludcmp
    INTEGER i,its,j,indx(NP)
    REAL d,den,f,fold,stpmax,sum,temp,test,fjac(NP,NP),g(NP),p(NP),
   *xold(NP),fmin
    EXTERNAL fmin
    nn=n
    f=fmin(x)
    test=0.
    do 11 i=1,n
     if(abs(fvec(i)).gt.test)test=abs(fvec(i))
11   continue
    if(test.lt..01*TOLF)return
    sum=0.
    do 12 i=1,n
     sum=sum+x(i)*x(i)
12   continue
    stpmax=STPMX*max(sqrt(sum),float(n))
    do 21 its=1,MAXITS
     call fdjac(n,x,fvec,NP,fjac)
     do 14 i=1,n
      sum=0.
      do 13 j=1,n
       sum=sum+fjac(j,i)*fvec(j)
13     continue
      g(i)=sum
14    continue
     do 15 i=1,n
      xold(i)=x(i)
15    continue
     fold=f
     do 16 i=1,n
      p(i)=-fvec(i)
16    continue
     call ludcmp(fjac,n,NP,indx,d)
```

```
      call lubksb(fjac,n,NP,indx,p)
      call lnsrch(n,xold,fold,g,p,x,f,stpmax,check,fmin)
      test=0.
      do 17 i=1,n
        if(abs(fvec(i)).gt.test)test=abs(fvec(i))
17    continue
      if(test.lt.TOLF)then
        check=.false.
        return
      endif
      if(check)then
        test=0.
        den=max(f,.5*n)
        do 18 i=1,n
          temp=abs(g(i))*max(abs(x(i)),1.)/den
          if(temp.gt.test)test=temp
18      continue
        if(test.lt.TOLMIN)then
          check=.true.
        else
          check=.false.
        endif
        return
      endif
      test=0.
      do 19 i=1,n
        temp=(abs(x(i)-xold(i)))/max(abs(x(i)),1.)
        if(temp.gt.test)test=temp
19    continue
      if(test.lt.TOLX)return
21    continue
      pause 'MAXITS exceeded in newt'
      END
C  (C) Copr. 1986-92 Numerical Recipes Software 0@>.1y.
      SUBROUTINE fdjac(n,x,fvec,np,df)
      INTEGER n,np,NMAX
      REAL df(np,np),fvec(n),x(n),EPS
      PARAMETER (NMAX=40,EPS=1.e-4)
CU    USES funcv
      INTEGER i,j
      REAL h,temp,f(NMAX)
      do 12 j=1,n
        temp=x(j)
        h=EPS*abs(temp)
        if(h.eq.0.)h=EPS
        x(j)=temp+h
        h=x(j)-temp
        call funcv(n,x,f)
```

```
      x(j)=temp
      do 11 i=1,n
       df(i,j)=(f(i)-fvec(i))/h
11     continue
12    continue
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software 0@>.1y.
      FUNCTION fmin(x)
      INTEGER n,NP
      REAL fmin,x(*),fvec
      PARAMETER (NP=40)
      COMMON /newtv/ fvec(NP),n
      SAVE /newtv/
CU    USES funcv
      INTEGER i
      REAL sum
      call funcv(n,x,fvec)
      sum=0.
      do 11 i=1,n
       sum=sum+fvec(i)**2
11    continue
      fmin=0.5*sum
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software 0@>.1y.
      SUBROUTINE lnsrch(n,xold,fold,g,p,x,f,stpmax,check,func)
      INTEGER n
      LOGICAL check
      REAL f,fold,stpmax,g(n),p(n),x(n),xold(n),func,ALF,TOLX
      PARAMETER (ALF=1.e-4,TOLX=1.e-7)
      EXTERNAL func
CU    USES func
      INTEGER i
      REAL a,alam,alam2,alamin,b,disc,f2,fold2,rhs1,rhs2,slope,sum,temp,
     *test,tmplam
      check=.false.
      sum=0.
      do 11 i=1,n
       sum=sum+p(i)*p(i)
11    continue
      sum=sqrt(sum)
      if(sum.gt.stpmax)then
       do 12 i=1,n
        p(i)=p(i)*stpmax/sum
12     continue
      endif
      slope=0.
```

```
      do 13 i=1,n
        slope=slope+g(i)*p(i)
13    continue
      test=0.
      do 14 i=1,n
        temp=abs(p(i))/max(abs(xold(i)),1.)
        if(temp.gt.test)test=temp
14    continue
      alamin=TOLX/test
      alam=1.
1     continue
        do 15 i=1,n
          x(i)=xold(i)+alam*p(i)
15      continue
        f=func(x)
        if(alam.lt.alamin)then
          do 16 i=1,n
            x(i)=xold(i)
16        continue
          check=.true.
          return
        else if(f.le.fold+ALF*alam*slope)then
          return
        else
          if(alam.eq.1.)then
            tmplam=-slope/(2.*(f-fold-slope))
          else
            rhs1=f-fold-alam*slope
            rhs2=f2-fold2-alam2*slope
            a=(rhs1/alam**2-rhs2/alam2**2)/(alam-alam2)
            b=(-alam2*rhs1/alam**2+alam*rhs2/alam2**2)/(alam-alam2)
            if(a.eq.0.)then
              tmplam=-slope/(2.*b)
            else
              disc=b*b-3.*a*slope
              tmplam=(-b+sqrt(disc))/(3.*a)
            endif
            if(tmplam.gt..5*alam)tmplam=.5*alam
          endif
        endif
        alam2=alam
        f2=f
        fold2=fold
        alam=max(tmplam,.1*alam)
      goto 1
      END
C (C) Copr. 1986-92 Numerical Recipes Software 0@>.1y.
      SUBROUTINE lubksb(a,n,np,indx,b)
```

```
      INTEGER n,np,indx(n)
      REAL a(np,np),b(n)
      INTEGER i,ii,j,ll
      REAL sum
      ii=0
      do 12 i=1,n
       ll=indx(i)
       sum=b(ll)
       b(ll)=b(i)
       if (ii.ne.0)then
        do 11 j=ii,i-1
         sum=sum-a(i,j)*b(j)
11      continue
       else if (sum.ne.0.) then
        ii=i
       endif
       b(i)=sum
12    continue
      do 14 i=n,1,-1
       sum=b(i)
       do 13 j=i+1,n
        sum=sum-a(i,j)*b(j)
13     continue
       b(i)=sum/a(i,i)
14    continue
      return
      END
C (C) Copr. 1986-92 Numerical Recipes Software 0@>.1y.
      SUBROUTINE ludcmp(a,n,np,indx,d)
      INTEGER n,np,indx(n),NMAX
      REAL d,a(np,np),TINY
      PARAMETER (NMAX=500,TINY=1.0e-20)
      INTEGER i,imax,j,k
      REAL aamax,dum,sum,vv(NMAX)
      d=1.
      do 12 i=1,n
       aamax=0.
       do 11 j=1,n
        if (abs(a(i,j)).gt.aamax) aamax=abs(a(i,j))
11     continue
       if (aamax.eq.0.) pause 'singular matrix in ludcmp'
       vv(i)=1./aamax
12    continue
      do 19 j=1,n
       do 14 i=1,j-1
        sum=a(i,j)
        do 13 k=1,i-1
         sum=sum-a(i,k)*a(k,j)
```

```
13      continue
        a(i,j)=sum
14    continue
      aamax=0.
      do 16 i=j,n
       sum=a(i,j)
       do 15 k=1,j-1
        sum=sum-a(i,k)*a(k,j)
15      continue
       a(i,j)=sum
       dum=vv(i)*abs(sum)
       if (dum.ge.aamax) then
        imax=i
        aamax=dum
       endif
16    continue
      if (j.ne.imax)then
       do 17 k=1,n
        dum=a(imax,k)
        a(imax,k)=a(j,k)
        a(j,k)=dum
17      continue
       d=-d
       vv(imax)=vv(j)
      endif
      indx(j)=imax
      if(a(j,j).eq.0.)a(j,j)=TINY
      if(j.ne.n)then
       dum=1./a(j,j)
       do 18 i=j+1,n
        a(i,j)=a(i,j)*dum
18      continue
      endif
19    continue
     return
     END
C  (C) Copr. 1986-92 Numerical Recipes Software 0@>.1y.
     SUBROUTINE broydn(x,n,check)
     INTEGER n,nn,NP,MAXITS
     REAL x(n),fvec,EPS,TOLF,TOLMIN,TOLX,STPMX
     LOGICAL check
     PARAMETER (NP=40,MAXITS=500,EPS=1.e-7,TOLF=1.e-4,TOLMIN=1.e-6,
    *TOLX=EPS,STPMX=.005)
     COMMON /newtv/ fvec(NP),nn
CU    USES fdjac,fmin,lnsrch,qrdcmp,qrupdt,rsolv
     INTEGER i,its,j,k
     REAL den,f,fold,stpmax,sum,temp,test,c(NP),d(NP),fvcold(NP),g(NP),
    *p(NP),qt(NP,NP),r(NP,NP),s(NP),t(NP),w(NP),xold(NP),fmin
```

```
      LOGICAL restrt,sing,skip
      EXTERNAL fmin
      nn=n
      f=fmin(x)
      test=0.
      do 11 i=1,n
        if(abs(fvec(i)).gt.test)test=abs(fvec(i))
11    continue
      if(test.lt..01*TOLF)return
      sum=0.
      do 12 i=1,n
        sum=sum+x(i)**2
12    continue
      stpmax=STPMX*max(sqrt(sum),float(n))
      restrt=.true.
      do 44 its=1,MAXITS
        if(restrt)then
          call fdjac(n,x,fvec,NP,r)
          call qrdcmp(r,n,NP,c,d,sing)
          if(sing) pause 'singular Jacobian in broydn'
          do 14 i=1,n
            do 13 j=1,n
              qt(i,j)=0.
13          continue
            qt(i,i)=1.
14        continue
          do 18 k=1,n-1
            if(c(k).ne.0.)then
              do 17 j=1,n
                sum=0.
                do 15 i=k,n
                  sum=sum+r(i,k)*qt(i,j)
15                continue
                sum=sum/c(k)
                do 16 i=k,n
                  qt(i,j)=qt(i,j)-sum*r(i,k)
16                continue
17              continue
            endif
18        continue
          do 21 i=1,n
            r(i,i)=d(i)
            do 19 j=1,i-1
              r(i,j)=0.
19          continue
21        continue
        else
          do 22 i=1,n
```

```
        s(i)=x(i)-xold(i)
22      continue
       do 24 i=1,n
        sum=0.
        do 23 j=i,n
         sum=sum+r(i,j)*s(j)
23      continue
        t(i)=sum
24      continue
       skip=.true.
       do 26 i=1,n
        sum=0.
        do 25 j=1,n
         sum=sum+qt(j,i)*t(j)
25      continue
        w(i)=fvec(i)-fvcold(i)-sum
        if(abs(w(i)).ge.EPS*(abs(fvec(i))+abs(fvcold(i))))then
         skip=.false.
        else
         w(i)=0.
        endif
26      continue
       if(.not.skip)then
        do 28 i=1,n
         sum=0.
         do 27 j=1,n
          sum=sum+qt(i,j)*w(j)
27       continue
         t(i)=sum
28      continue
        den=0.
        do 29 i=1,n
         den=den+s(i)**2
29      continue
        do 31 i=1,n
         s(i)=s(i)/den
31      continue
        call qrupdt(r,qt,n,NP,t,s)
        do 32 i=1,n
         if(r(i,i).eq.0.) pause 'r singular in broydn'
         d(i)=r(i,i)
32      continue
       endif
      endif
      do 34 i=1,n
       sum=0.
       do 33 j=1,n
        sum=sum+qt(i,j)*fvec(j)
```

```
33      continue
        g(i)=sum
34      continue
      do 36 i=n,1,-1
        sum=0.
        do 35 j=1,i
          sum=sum+r(j,i)*g(j)
35      continue
        g(i)=sum
36      continue
      do 37 i=1,n
        xold(i)=x(i)
        fvcold(i)=fvec(i)
37      continue
      fold=f
      do 39 i=1,n
        sum=0.
        do 38 j=1,n
          sum=sum+qt(i,j)*fvec(j)
38      continue
        p(i)=-sum
39      continue
      call rsolv(r,n,NP,d,p)
      call lnsrch(n,xold,fold,g,p,x,f,stpmax,check,fmin)
      test=0.
      do 41 i=1,n
        if(abs(fvec(i)).gt.test)test=abs(fvec(i))
41      continue
      if(test.lt.TOLF)then
        check=.false.
        return
      endif
      if(check)then
        if(restrt)then
          return
        else
          test=0.
          den=max(f,.5*n)
          do 42 i=1,n
            temp=abs(g(i))*max(abs(x(i)),1.)/den
            if(temp.gt.test)test=temp
42        continue
          if(test.lt.TOLMIN)then
            return
          else
            restrt=.true.
          endif
        endif
```

```
      else
        restrt=.false.
        test=0.
        do 43 i=1,n
          temp=(abs(x(i)-xold(i)))/max(abs(x(i)),1.)
          if(temp.gt.test)test=temp
43        continue
        if(test.lt.TOLX)return
      endif
44    continue
      pause 'MAXITS exceeded in broydn'
      END
C (C) Copr. 1986-92 Numerical Recipes Software 0@>.1y.
      SUBROUTINE rotate(r,qt,n,np,i,a,b)
      INTEGER n,np,i
      REAL a,b,r(np,np),qt(np,np)
      INTEGER j
      REAL c,fact,s,w,y
      if(a.eq.0.)then
        c=0.
        s=sign(1.,b)
      else if(abs(a).gt.abs(b))then
        fact=b/a
        c=sign(1./sqrt(1.+fact**2),a)
        s=fact*c
      else
        fact=a/b
        s=sign(1./sqrt(1.+fact**2),b)
        c=fact*s
      endif
      do 11 j=i,n
        y=r(i,j)
        w=r(i+1,j)
        r(i,j)=c*y-s*w
        r(i+1,j)=s*y+c*w
11    continue
      do 12 j=1,n
        y=qt(i,j)
        w=qt(i+1,j)
        qt(i,j)=c*y-s*w
        qt(i+1,j)=s*y+c*w
12    continue
      return
      END
C (C) Copr. 1986-92 Numerical Recipes Software 0@>.1y.
      SUBROUTINE rsolv(a,n,np,d,b)
      INTEGER n,np
      REAL a(np,np),b(n),d(n)
```

```
      INTEGER i,j
      REAL sum
      b(n)=b(n)/d(n)
      do 12 i=n-1,1,-1
        sum=0.
        do 11 j=i+1,n
          sum=sum+a(i,j)*b(j)
11      continue
        b(i)=(b(i)-sum)/d(i)
12    continue
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software 0@>.1y.
      SUBROUTINE qrdcmp(a,n,np,c,d,sing)
      INTEGER n,np
      REAL a(np,np),c(n),d(n)
      LOGICAL sing
      INTEGER i,j,k
      REAL scale,sigma,sum,tau
      sing=.false.
      scale=0.
      do 17 k=1,n-1
        do 11 i=k,n
          scale=max(scale,abs(a(i,k)))
11      continue
        if(scale.eq.0.)then
          sing=.true.
          c(k)=0.
          d(k)=0.
        else
          do 12 i=k,n
            a(i,k)=a(i,k)/scale
12        continue
          sum=0.
          do 13 i=k,n
            sum=sum+a(i,k)**2
13        continue
          sigma=sign(sqrt(sum),a(k,k))
          a(k,k)=a(k,k)+sigma
          c(k)=sigma*a(k,k)
          d(k)=-scale*sigma
          do 16 j=k+1,n
            sum=0.
            do 14 i=k,n
              sum=sum+a(i,k)*a(i,j)
14          continue
            tau=sum/c(k)
            do 15 i=k,n
```

```
          a(i,j)=a(i,j)-tau*a(i,k)
15        continue
16       continue
       endif
17    continue
      d(n)=a(n,n)
      if(d(n).eq.0.)sing=.true.
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software 0@>.1y.
      SUBROUTINE qrupdt(r,qt,n,np,u,v)
      INTEGER n,np
      REAL r(np,np),qt(np,np),u(np),v(np)
CU    USES rotate
      INTEGER i,j,k
      do 11 k=n,1,-1
       if(u(k).ne.0.)goto 1
11    continue
      k=1
1     do 12 i=k-1,1,-1
       call rotate(r,qt,n,np,i,u(i),-u(i+1))
       if(u(i).eq.0.)then
         u(i)=abs(u(i+1))
       else if(abs(u(i)).gt.abs(u(i+1)))then
         u(i)=abs(u(i))*sqrt(1.+(u(i+1)/u(i))**2)
       else
         u(i)=abs(u(i+1))*sqrt(1.+(u(i)/u(i+1))**2)
       endif
12    continue
      do 13 j=1,n
       r(1,j)=r(1,j)+u(1)*v(j)
13    continue
      do 14 i=1,k-1
       call rotate(r,qt,n,np,i,r(i,i),-r(i+1,i))
14    continue
      return
      END


**********************************************************************
          SUBROUTINE amoeba(p,y,mp,np,ndim,ftol,funk,iter)
          INTEGER iter,mp,ndim,np,NMAX,ITMAX
          REAL ftol,p(mp,np),y(mp),funk
          PARAMETER (NMAX=20,ITMAX=90000)
           EXTERNAL funk
CU    USES amotry,funk
          INTEGER i,ihi,ilo,inhi,j,m,n
```

```
      REAL rtol,sum,swap,ysave,ytry,psum(NMAX),amotry
       iter=0
1     do 12 n=1,ndim
      sum=0.
      do 11 m=1,ndim+1
        sum=sum+p(m,n)
11    continue
      psum(n)=sum
12   continue
2    ilo=1
   if (y(1).gt.y(2)) then
      ihi=1
      inhi=2
   else
      ihi=2
      inhi=1
   endif
   do 13 i=1,ndim+1
      if(y(i).le.y(ilo)) ilo=i
      if(y(i).gt.y(ihi)) then
        inhi=ihi
        ihi=i
      else if(y(i).gt.y(inhi)) then
        if(i.ne.ihi) inhi=i
      endif
13   continue
   rtol=2.*abs(y(ihi)-y(ilo))/(abs(y(ihi))+abs(y(ilo)))
   if (rtol.lt.ftol) then
      swap=y(1)
      y(1)=y(ilo)
      y(ilo)=swap
      do 14 n=1,ndim
        swap=p(1,n)
        p(1,n)=p(ilo,n)
        p(ilo,n)=swap
14     continue
      return
   endif
   if (iter.ge.ITMAX) pause 'ITMAX exceeded in amoeba'
   iter=iter+2
   ytry=amotry(p,y,psum,mp,np,ndim,funk,ihi,-1.0)
   if (ytry.le.y(ilo)) then
      ytry=amotry(p,y,psum,mp,np,ndim,funk,ihi,2.0)
   else if (ytry.ge.y(inhi)) then
      ysave=y(ihi)
      ytry=amotry(p,y,psum,mp,np,ndim,funk,ihi,0.5)
      if (ytry.ge.ysave) then
        do 16 i=1,ndim+1
```

```
            if(i.ne.ilo)then
              do 15 j=1,ndim
                   psum(j)=0.5*(p(i,j)+p(ilo,j))
                   p(i,j)=psum(j)
15          continue
              y(i)=funk(psum)
            endif
16        continue
            iter=iter+ndim
            goto 1
          endif
      else
          iter=iter-1
      endif
      goto 2
      END
C  (C) Copr. 1986-92 Numerical Recipes Software 0@>.1y.
      FUNCTION amotry(p,y,psum,mp,np,ndim,funk,ihi,fac)
      INTEGER ihi,mp,ndim,np,NMAX
      REAL amotry,fac,p(mp,np),psum(np),y(mp),funk
      PARAMETER (NMAX=20)
      EXTERNAL funk
CU    USES funk
      INTEGER j
      REAL fac1,fac2,ytry,ptry(NMAX)
      fac1=(1.-fac)/ndim
      fac2=fac1-fac
      do 11 j=1,ndim
          ptry(j)=psum(j)*fac1-p(ihi,j)*fac2
11    continue
      ytry=funk(ptry)
      if (ytry.lt.y(ihi)) then
          y(ihi)=ytry
          do 12 j=1,ndim
            psum(j)=psum(j)-p(ihi,j)+ptry(j)
            p(ihi,j)=ptry(j)
12      continue
      endif
      amotry=ytry
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software 0@>.1y.
*****************************************************************
          real function funk(x)
          implicit double precision (a-h,o-z)
          double precision dT,qual
          real x(2),funk
          real xnewt(2)
```

```
      real xnewtpass(2)
      logical check
      character*70 config,geometry,fluid,solid
      double precision delem(100),heatload(100),trgh(100),Tleft(100)
     * ,ratio(100),aspect(100),poros(100)
      INTEGER NUMLAY(100),NUMSEC
      common /switchs/ geometry,fluid,solid,config
      common /account/ numtot
      common /xnewtpass/ xnewtpass
      COMMON /VARS2/ NUMLAY,NUMSEC
      COMMON /VARS3/ DELEM,HEATLOAD,TRGH,TLEFT,RATIO,
     *  ASPECT,POROS,PAMB,p1not
! Set the new values of right and left end resonator tube lengths.
      delem(1)=real(x(1))
      delem(numsec)=real(x(2))
      totalRadius=0.d0
      do 107 i=1,numsec
107   totalRadius=totalRadius+delem(i)
      ratio(1)=totalRadius/2.d0 ! Constant radius to height ratio=2:1
      ratio(numsec)=ratio(1)
      nnewt=2 ! Two variables in the newton raphson itteration.
      pi=4.d0*datan(1.d0)
! Just use the previous values of Xnewt from tae, as per the common.
*     call TAE(f0est,dT,qual) ! Estimate for frequency,dT,qual.
*     xnewt(1)=real(f0est) ! Resonant frequency.
*     if (config.eq.'quality') then
*      xnewt(2)=real(pi*f0est/qual) ! W has an imaginary part.
*     else
*      xnewt(2)=real(dT) ! W is real, prime mover.
*     end if
      xnewt(1)=xnewtpass(1)
      xnewt(2)=xnewtpass(2)
      write(*,*) ' xnewt(1)=resfre before calling newt',xnewt(1)
      call newt(xnewt,nnewt,check)
      if (config.eq.'quality') then
       qual=pi*real(xnewt(1)/xnewt(2))
       write(*,*) 'QUALITY FACTOR =',real(qual)
       funk = -1./real(qual)
      else if (config.eq.'wallend') then
       funk = -real(Trgh(numsec)-Trgh(1))/1.e5
      else if (config.eq.'center') then
       funk = -real(Trgh(1)-Trgh(numsec))/1.e5
      end if
      write(*,*) 'funk =',funk
      write(*,*) 'Resonance frequency = ',xnewt(1)
      return
      end
```

```
          xnewt(1)=real(f0est) ! Resonant frequency.
          if (config.eq.'quality') then
           xnewt(2)=real(pi*f0est/qual) ! W has an imaginary part.
          else
           xnewt(2)=real(dT) ! W is real, prime mover.
          end if
! Compute the system for the input file.
          call newt(xnewt,nnewt,check)
! Write out the computed frequency, and Q or onset dT.
          fonset=dreal(xnewt(1))
          write(*,*) ' Resonant frequency accurate=',xnewt(1),' Hz'
          if (config.eq.'quality') then
           qual=pi*fonset/dreal(xnewt(2))
           write(*,*) ' Quality factor=',real(qual)
          else
           Tonset=dreal(xnewt(2))
          end if
          return
          end
*************************************************************************
* Storevalues - makes an .params file after optimization.
*************************************************************************
      SUBROUTINE storevalues
          implicit double precision (a-h,o-z)
* VARIABLES WHICH DEFINE THE TAE.
      CHARACTER*70 SECTYP(100),ETYPE(100),TERMINR,TERMINL
      INTEGER NUMLAY(100),NUMSEC
      double precision DELEM(100),TRGH(100),TLEFT(100),RATIO(100)
     *,ASPECT(100),POROS(100),HEATLOAD(100)
      INTEGER J
      CHARACTER DUMMY
          character*70 geometry,fluid,solid,config
          character*80 line
          common /switchs/ geometry,fluid,solid,config
      COMMON /VARS1/ SECTYP,ETYPE,TERMINR,TERMINL
      COMMON /VARS2/ NUMLAY,NUMSEC
      COMMON /VARS3/ DELEM,HEATLOAD,TRGH,TLEFT,RATIO,
     *  ASPECT,POROS,PAMB,P1not
1     FORMAT (A1)
2     FORMAT (A70)
3     FORMAT (A80)
      REWIND 2
          call system('cls')
1011    call system('dir /b *.params')
          write(*,*) 'Enter filename :  '
          read(*,3) line
          open(3,file=line,status='new',err=1010)
```

```
      do 300 j=1,24
      READ (2,3) line
300   write(3,3) line
            read (2,1) dummy
            write (3,2) fluid
   READ (2,3) line
   write (3,3) line
            read (2,1) dummy
            write (3,2) solid
   READ (2,3) line
      write(3,3) line
   READ (2,3) line
      write(3,3) line
            READ (2,1) dummy
            write (3,2) TERMINR
   READ (2,3) line
      write(3,3) line
   READ (2,3) line
      write(3,3) line
            READ (2,1) dummy
            write (3,2) TERMINL
   READ (2,3) line
      write(3,3) line
   READ (2,3) line
      write(3,3) line
            READ (2,1) dummy
            write(3,*) PAMB,P1not
   READ (2,3) line
      write(3,3) line
   READ (2,3) line
      write(3,3) line
            READ (2,1) dummy
            write (3,*) NUMSEC
      DO 10 J=NUMSEC,1,-1
      write(3,*)
   * '******** DEFINITION OF SECTION',j,'*****************'
      write(3,*)
   * 'SECTION TYPE, ONE OF OPENTU, HOLE, HEXCH, OR STACK.'
      write (3,2) SECTYP(J)
      write(3,*)
   * 'Heat load on the element in case it is a heat exchanger.'
      write(3,*)
   * 'Give the number in Watts.'
      write (3,*) HEATLOAD(J)
      write(3,*)
   * 'NUMBER OF LAYERS THIS SECTION IS BROKEN UP INTO.'
      write(3,*)
   * '1<= NUMLAY <= 100 PRACTICALLY.'
```

```
      write(3,*) NUMLAY(J)
      if (sectyp(j).eq.'HOLE') then
      write(3,*)
     * 'DIAMETER OF HOLE which is also the length of section.'
      else
      write(3,*)
     * 'LENGTH OF THE SECTION.'
      end if
      write(3,*)
     * 'METERS.'
      write(3,*) DELEM(J)
      write(3,*)
     * 'TEMPERATURE OF RGH END OF SECTION. FOR AN ISOTHERMAL SECTION'
      write(3,*)
     * 'SUCH AS OPEN TUBE OR HEAT EXCH, USE TRGH = TLEFT. KELVIN.'
      write(3,*) TRGH(J)
      write(3,*)
     * 'TEMPERATURE OF LEFT END OF SECTION. FOR AN ISOTHERMAL SECTION'
      write(3,*)
     * 'SUCH AS OPEN TUBE OR HEAT EXCH, USE TRGH = TLEFT. KELVIN.'
      write(3,*) TLEFT(J)
      if (sectyp(j).eq.'HOLE') then
      write(3,*)
     * 'LENGTH of tube connected to the hole.'
      write(3,*)
     * 'IS just the plate thickness for a hole with no tube on it.'
      else
      write(3,*)
     *'RATIO OF 2 PoreArea TO PORE PERIM (M). FOR:CYL=RADIUS,SLIT='
      write(3,*)
     *'WIDTH, RECT=2SwA/(1+A) A>1=SidesAspRatio,Sw=Shortestsemiwidth.'
      end if
      write(3,*) RATIO(J)
      write(3,*)
     * 'ASPECT RATIO OF THE PORE. VALID FOR RECTANGULAR PORES ONLY.'
      write(3,*)
     * 'NECESSARY AS A GENERAL RULE.'
      write(3,*) ASPECT(J)
      write(3,*)
     * 'POROSITY OF THE SECTION.  FOR OPEN TUBE, USE POROSITY = 1.'
      write(3,*)
     * 'FOR OTHER TYPES OF SECTIONS, POROSITY <=1.'
      write(3,*) POROS(J)
      write(3,*)
     * 'END OF SECTION 'j,' ******************************'
10    continue
     REWIND 2
      close(3)
```

```
      RETURN
1010  write(*,*) ' THE FILE MUST BE NEW, TRY AGAIN'
      goto 1011
      END
```

Appendix B

Transport Coefficients

The formulation of the transport coefficients of viscosity and thermal conductivity for a ν component gas is presented in this appendix. This is based on a collection of the work of Enskog[31], Chapman[29], and Hirschfelder[28].

B.1 Transport Theory Introduction

The kinetic theory of gases is based upon the knowledge of the molecular distribution function, $f(\boldsymbol{r},\boldsymbol{v},t)$. This function represents the probable number of molecules at time t which lie in a unit volume about point $\boldsymbol{r}$ and have velocities with a unit range about $\boldsymbol{v}$. If there are no gradients in the composition, velocity and temperature in the gas then $f(\boldsymbol{r},\boldsymbol{v},t)$ reduces to the Maxwellian distribution

$$f^{[0]} = n\left(\frac{m}{2\pi kT}\right)^{3/2} \exp\left(-\frac{mv^2}{2kT}\right). \qquad \text{B.1}$$

Where n is the number of molecules, m is the mass, k is Boltzmann's constant, T is the temperature, and v is the velocity.

A gas in non-equilibrium conditions, gradients exists in one or more of the macroscopic properties. These gradients are the cause of the molecular transport of

momentum and thermal energy and are represented by the transport coefficients of viscosity and thermal conductivity respectively. The distribution function then satisfies

$$\frac{\partial_e f}{\partial t} = \frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \frac{\partial f}{\partial \boldsymbol{r}} + \boldsymbol{F} \cdot \frac{\partial f}{\partial \boldsymbol{v}}. \qquad \text{B.2}$$

This is the Boltzmann integro-differential equation. $\partial_e f/\partial t$ is the rate of change in the distribution function due to the collisions between molecules. Vectors $\boldsymbol{r}$ and $\boldsymbol{v}$ are the position and velocity of the molecules and $\boldsymbol{F}$ is the external force action on the molecules. In the limit where the properties of the gases under consideration is nearly Maxwellian, the Boltzmann equation can be solved by a perturbation method developed by Chapmann[29] and independently by Enskog[31].

Enskog[31] in his series solution for the Boltzmann equation (Equation B.2) identifies a perturbation function

$$\Phi_i = -\left(\boldsymbol{A} \cdot \frac{\partial nT}{\partial \boldsymbol{r}}\right) - \left(\boldsymbol{B} \cdot \left(\boldsymbol{B} \cdot \frac{\partial \boldsymbol{v}_0}{\partial \boldsymbol{r}}\right)\right) + n\sum_j \left(C_i^j \cdot \boldsymbol{d}_i\right). \qquad \text{B.3}$$

The functions ***A***, ***B***, ***C***, and ***d*** (the functional form of each term will be defined in the derivation of the different transport properties) are all functions of the composition, temperature, and reduced velocity,

$$W_j = \sqrt{\frac{m_j}{2kT}} V_j . \qquad \text{B.4}$$

The mean velocity of a molecule at position ***r*** and time t can be written as

$$u(r,t) = \langle v(r,t) \rangle = \frac{1}{n(r,t)} \int f(r,v,t) v(r,t) d^3 v \qquad \text{B.5}$$

which is the average velocity (hydronamic velocity) at point ***r***. The peculiar velocity, which is the velocity of a molecule with respect to a reference frame moving at $u(r,t)$ and is defined as $V(r,t) = v(r,t) - u(r,t)$.

The resulting perturbation function (Equation B.3) can then be used to obtain the transport coefficients of viscosity and thermal conductivity in terms of a set of integrals[29],

$$\Omega_{ij}^{(l,s)} = \sqrt{\frac{2\pi kT}{\mu_{ij}}} \int_0^\infty \int_0^\infty e^{-\gamma_{ij}^2} \gamma_{ij}^{2s+3} \left(1 + \cos^l \chi\right) b db d\gamma_{ij} . \qquad \text{B.6}$$

Where $\gamma_{ij} = mg^2/2kT$ is the reduced initial relative velocity. These integrals involve explicitly the dynamics of a molecular encounter through the reduced mass, μ_{ij} ,of the colliding molecules, the impact parameter, b, and the angle of deflection,

$$\chi(g,b) = \pi - 2b\int_r^\infty \frac{dr/r^2}{\sqrt{1-\frac{b^2}{r^2}-\frac{\varphi(r)}{\mu g^2/2}}}. \qquad \text{B.7}$$

The intermolecular potential used in the calculations of the various transport properties is

$$\varphi(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \qquad \text{B.8}$$

which is frequently referred to as the Lennard-Jones potential. σ is the value of r in which $\varphi(r)=0$, ε is the maximum energy of attraction which occurs at $r = 2^{1/6}\sigma$.

The mechanism of the transport of each molecular property can be treated similarly and is denoted by ϕ. The transport property can then be written as

$$\Psi_j(\boldsymbol{r},\boldsymbol{v},t) = \int f_j \phi_j \boldsymbol{V}_j d\boldsymbol{V}_j \qquad \text{B.9}$$

and is called the flux tensor associated with the transport property ϕ. This tensor has the physical significance that the component of the tensor in any direction is the flux

across a surface normal to that direction. The non-equilibrium distribution function is denoted by

$$f_j = f_j^{[0]}(1+\Phi_j). \qquad \text{B.10}$$

B.2 Viscosity

The viscous coefficient arises through the transport property of momentum. The stress tensor $\underline{\mathbf{P}}$ and can be written using $\phi_j = m_j V_j$ and f_j in Equation B.9

$$\underline{\mathbf{P}} = \sum_j m_j \left\{ \int f_j^{[0]} V_j V_j dV + \int f_j^{[0]} \Phi_j V_j V_j dV \right\}. \qquad \text{B.11}$$

The first integral can be evaluated directly and substituting the perturbation function, Equation B.2, in the second integral the stress tensor becomes

$$\mathbf{P} = pV - \sum_j m_j \left\{ \int f_j^{[0]} \left(B_j \cdot \left(B_j \cdot \frac{\partial v_0}{\partial r} \right) \right) V_j V_j dV \right\}. \qquad \text{B.12}$$

The perturbation function terms with A_j and C_j are not shown since they do not contribute due to symmetry. The first term is the equilibrium hydrostatic pressure at

the local temperature and density with p = nkT. $\boldsymbol{B}_j$ is a symmetrical non-divergent tensor and is of the form $\boldsymbol{B}_j = \{\mathbf{W}_j \mathbf{W}_j - 1/3\ W_j^2\ U\}B_j(W_j)$. Expanding the dot products and with some symmetry arguments the stress tensor reduces to

$$\mathbf{P} = p\mathbf{V} - \frac{2}{15}\sum_j \frac{m_j^2}{2kT}\left[\int f_j^{[0]} V_j^4 B_j(W_j) d\boldsymbol{V}\right]\mathbf{S}, \qquad \text{B.13}$$

where S is the rate of shear tensor, defined by

$$\mathbf{S} = \frac{1}{2}\left[\frac{\partial v_{0\beta}}{\partial x_\alpha} + \frac{\partial v_{0\alpha}}{\partial x_\beta}\right] - \frac{1}{3}\delta_{\alpha\beta}\left(\frac{\partial}{\partial \boldsymbol{r}} \cdot \boldsymbol{v}_0\right). \qquad \text{B.14}$$

The definition of the coefficient of viscosity is given by the relation

$$\mathbf{P} = p\mathbf{U} - 2\eta\mathbf{S} \qquad \text{B.15}$$

comparing Equation B.13 and B.14 yields the coefficient of viscosity

$$\eta = \frac{1}{15}\sum_j \frac{m_j^2}{2kT}\left[\int f_j^{[0]} V_j^4 B_j(W_j) d\boldsymbol{V}\right]. \qquad \text{B.16}$$

The Sonine polynomial expansion presented in appendix, C Equation C.5 can be used to rewrite the viscous coefficient as

$$\eta(\xi) = \frac{1}{15}\sum_j \frac{m_j^2}{2kT}\sum_{m=0}^{\xi-1} b_{jm}(\xi)\left[\int f_j^{[0]} V_j^4 S_{5/2}^m(W_j^2)dV_j\right]. \qquad \text{B.17}$$

The argument of $\eta(\xi)$ indicates the order of approximation. Employing orthogonality relation of the Sonine polynomials (defined in Appendix C equation C.4) the coefficient of viscosity in terms of the Sonine expansion coefficient is

$$\eta(\xi) = \frac{1}{2}kT\sum_j n_j b_{j0}(\xi). \qquad \text{B.18}$$

The first approximation to the coefficient of viscosity for a ν component mixture can be written as

$$\eta(1) = \frac{1}{2}kT\sum_j n_j b_{j0}(1) \qquad \text{B.19}$$

where the $b_{j0}(1)$ are determined by ν equations of the form

$$\sum_j \left(\frac{Q_{ij}^{00}}{R_{i0}} \right) b_{j0}(1) = -1 \qquad i = 1, 2, 3, \cdots, \nu \qquad \text{B.20}$$

with

$$Q_{ij}^{00} = \sum_m n_i n_m \left(\delta_{ij} [W_i; W_i]_{im} + \delta_{jm} [W_i; W_m]_{im} \right), \qquad \text{B.21}$$

$W_i = W_i W_i - 1/3 W_i^2 V$, and $R_{i0} = -5n_i$. The $[W_i ; W_i]$ is defined in appendix C.

In Equation B.19, $b_{j0}(1)$ can be determined as a ratio of two determinants of order ν by Cramer's rule. However, the quantity $\Sigma_j n_j b_{j0}(1)$ must be written as a ratio of two determinants, the one in the numerator being of order $\nu+1$ and that in the denominator of order ν. The coefficient of viscosity can then be written as

$$\eta(1) = -\frac{\begin{vmatrix} H_{11} & \cdots & H_{1\nu} & n_1/n \\ \vdots & \vdots & \vdots & \vdots \\ H_{1\nu} & \cdots & H_{\nu\nu} & n_\nu/n \\ n_1/n & \cdots & n_\nu/n & 0 \end{vmatrix}}{|H_{ij}|}, \qquad \text{B.22}$$

with H_{ij} written in terms of $\Omega^{(l,s)}$ defined by Equation B.6 is

$$H_{ij} = \frac{32}{15}\frac{n_i m_i}{n^2 m_j kT}\sum_l \frac{n_l m_l}{(m_i + m_l)^2}\begin{bmatrix} 5m_j(\delta_{ij} - \delta_{jl})\Omega_{il}^{(1,1)} \\ +\frac{3}{2}m_l(\delta_{ij} + \delta_{jl})\Omega_{il}^{(2,2)} \end{bmatrix} \qquad \text{B.23}$$

B.3 Thermal conductivity

A similar procedure can be used to find the thermal conductivity through the transport property of energy. The use of the purtabation function and Equation B.9 with $\phi_j = 1/2\ m_j V_j^2$ the energy flux can be written as[28,29]

$$\mathbf{q}(1) = \frac{1}{2}\sum_j m_j \int\int f_j^{[0]} V_j^2 \boldsymbol{V}_j \left[\left(\boldsymbol{A}_j \cdot \frac{\partial \ln T}{\partial \boldsymbol{r}}\right) + n\sum_k \left(\boldsymbol{C}_j^k \cdot \boldsymbol{d}_k\right)\right] d\boldsymbol{V}_j \,. \qquad \text{B.24}$$

The term containing $\boldsymbol{B}_j$ does not contribute to the energy flux and is ignored. Substituting $\boldsymbol{A}_j = \mathbf{W}_j\ A_j(W_j)$, $\boldsymbol{C}_j^{(k)} = \mathbf{W}_j\ C_j^{(k)}(W_j)$ and keeping just the thermal flux terms

$$\boldsymbol{q}_T = -kT\sum_j \int \left\{ n\sum_k \begin{bmatrix} C_j^{(k)}(W_j)\left(\mathbf{W}_j \cdot \mathbf{d}_j\right) \\ -A_j(W_j)\left(\mathbf{W}_j \cdot \frac{\partial \ln T}{\partial \boldsymbol{r}}\right) \end{bmatrix} \right\} \left(\frac{5}{2} - W_j^2\right) \mathbf{V}_j F_j^{[0]} d\mathbf{V}_j , \qquad \text{B.25}$$

Expanding and using symmetry arguments it follows that

$$\mathbf{q}_T = -\lambda' \frac{\partial T}{\partial \mathbf{r}} - \frac{\sqrt{2}}{3} n \sqrt{(kT)^3} \sum_k \mathbf{d}_k \sum_j \frac{1}{\sqrt{m_j}} \int C_j^{(k)}(W_j)\left(\frac{5}{2} - W_j^2\right) W_j^2 F_j^{[0]} dV \qquad \text{B.26}$$

where

$$\lambda' = -\frac{\sqrt{2}}{3} k \sqrt{kT} \sum_j \frac{1}{\sqrt{m_j}} \int A_j(W_j)\left(\frac{5}{2} - W_j^2\right) W_j^2 F_j^{[0]} dV . \qquad \text{B.27}$$

Chapman's and Enskog's solution[29,30] to Equation B.26 is

$$\mathbf{q}_T = -\lambda' \frac{\partial T}{\partial \mathbf{r}} - nkT \sum_j \frac{1}{n_j m_j} D_j^T \mathbf{d}_j \qquad \text{B.28}$$

where

$$D_j^T = \frac{n_j m_j}{2} \sqrt{\frac{2kT}{m_j}} a_{j0} \qquad \text{B.29}$$

and

$$\mathbf{d}_j = \frac{\partial \ln T}{\partial \mathbf{r}} \sum_i \frac{n_i n_j}{n^2 \mathcal{D}_{ij}} \left(\frac{D_i^T}{n_i m_i} - \frac{D_i^T}{n_j m_j} \right) + \sum_i \frac{n_i n_j}{n^2 \mathcal{D}_{ij}} \left(\mathbf{V}_i - \mathbf{V}_j \right). \qquad \text{B.30}$$

The term $\mathcal{D}_{ij}$ is the coefficient of diffusion for a binary mixture and is

$$\mathcal{D}_{ij} = \frac{3\left(m_i + m_j\right)}{16 n m_i m_j} \frac{kT}{\Omega_{ij}^{(1,1)}}. \qquad \text{B.31}$$

The expression for the heat flux written in terms of the diffusion velocities and temperature gradient is

$$\mathbf{q}_T = -\lambda' \frac{\partial T}{\partial \mathbf{r}} + \frac{k}{2n} \sum_{i,j} \frac{n_i n_j}{\mathcal{D}_{ij}} \left(\frac{D_i^T}{n_i m_i} - \frac{D_i^T}{n_j m_j} \right)^2 \frac{\partial T}{\partial \mathbf{r}} \qquad \text{B.32}$$

using the definition $\mathbf{q}_T = -\lambda \, \partial T/\partial \mathbf{r}$ the coefficient of thermal conductivity can be expressed as

$$\lambda = -\lambda' + \frac{k}{2n} \sum_{i,j} \frac{n_i n_j}{\mathcal{D}_{ij}} \left(\frac{D_i^T}{n_i m_i} - \frac{D_i^T}{n_j m_j} \right)^2 \qquad \text{B.33}$$

where

$$\lambda' = \frac{5}{4} k \sum_j n_j \sqrt{\frac{2kT}{m_j}} a_{j1}. \qquad \text{B.34}$$

Using the same argument in deriving Equation B.19, λ' becomes

$$\lambda' = -\frac{75}{8} k^2 T \frac{\begin{vmatrix} q_{11}^{00} & \cdots & q_{1v}^{00} & q_{11}^{01} & \cdots & q_{1v}^{01} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ q_{v1}^{00} & \cdots & q_{vv}^{00} & q_{v1}^{01} & \cdots & q_{vv}^{01} & 0 \\ q_{11}^{10} & \cdots & q_{1v}^{10} & q_{11}^{11} & \cdots & q_{1v}^{11} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ q_{v1}^{10} & \cdots & q_{vv}^{10} & q_{v1}^{11} & \cdots & q_{vv}^{11} & 1 \\ 0 & \cdots & 0 & 1 & \cdots & 1 & 0 \end{vmatrix}}{\begin{vmatrix} q_{11}^{00} & \cdots & q_{1v}^{00} & q_{11}^{01} & \cdots & q_{1v}^{01} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ q_{v1}^{00} & \cdots & q_{vv}^{00} & q_{v1}^{01} & \cdots & q_{vv}^{01} \\ q_{11}^{10} & \cdots & q_{1v}^{10} & q_{11}^{11} & \cdots & q_{1v}^{11} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ q_{v1}^{10} & \cdots & q_{vv}^{10} & q_{v1}^{11} & \cdots & q_{vv}^{11} \end{vmatrix}} \qquad \text{B.35}$$

where

$$q_{ij}^{mm} = \frac{\sqrt{m_i m_j}}{n_i n_j} Q_{ij}^{mm'}, \qquad \text{B.36}$$

$$Q_{ij}^{00} = 8 \sum_k \frac{n_k m_k}{\sqrt{m_i m_j}(m_i + m_k)} \left[n_i m_i (\delta_{ij} - \delta_{jk}) - n_j m_j (1 - \delta_{ik}) \right] \Omega_{ik}^{(1,1)}, \qquad \text{B.37}$$

$$Q_{ij}^{01} = -8\left(\frac{m_i}{m_j}\right)^{3/2} \sum_k \frac{n_i n_k m_k^2}{(m_i + m_k)^2} (\delta_{ij} - \delta_{jk}) \left[\Omega_{ik}^{(1,2)} - \frac{5}{2} \Omega_{ik}^{(1,1)} \right], \qquad \text{B.38}$$

$$Q_{ij}^{10} = \frac{m_j}{m_i} Q_{ij}^{01}, \qquad \text{B.39}$$

and

$$Q_{ij}^{01} = 8\left(\frac{m_i}{m_j}\right)^{3/2} \sum_k \frac{n_i n_k m_k^2}{(m_i + m_k)^3} \left\{ \begin{array}{l} (\delta_{ij} - \delta_{jk}) \left[\begin{array}{l} \frac{5}{4}\left(6m_j^2 + 5m_k^2\right)\Omega_{ik}^{(1,1)} \\ -5m_k^2 \Omega_{ik}^{(1,2)} \\ +m_k^2 \Omega_{ik}^{(1,3)} \end{array} \right] \\ +(\delta_{ij} - \delta_{jk}) 2 m_j m_k \Omega_{ik}^{(2,2)} \end{array} \right\}. \qquad \text{B.40}$$

The Equation B.29 can be written as

$$D_i^T = n_i \sqrt{\frac{m_i kT}{2}} \frac{\begin{vmatrix} Q_{11}^{00} & Q_{12}^{00} & \cdots & Q_{1v}^{01} & Q_{11}^{01} & Q_{12}^{01} & \cdots & Q_{1v}^{01} & 0 \\ Q_{21}^{00} & Q_{22}^{00} & \cdots & Q_{2v}^{00} & Q_{21}^{01} & Q_{22}^{01} & \cdots & Q_{2v}^{01} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Q_{v1}^{00} & Q_{v2}^{00} & \cdots & Q_{vv}^{00} & Q_{v1}^{01} & Q_{v2}^{01} & \cdots & Q_{vv}^{01} & 0 \\ Q_{11}^{01} & Q_{12}^{01} & \cdots & Q_{1v}^{10} & Q_{11}^{11} & Q_{12}^{11} & \cdots & Q_{1v}^{11} & R_{11} \\ Q_{21}^{10} & Q_{22}^{10} & \cdots & Q_{2v}^{10} & Q_{21}^{11} & Q_{22}^{11} & \cdots & Q_{2v}^{11} & R_{21} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Q_{v1}^{10} & Q_{v2}^{10} & \cdots & Q_{vv}^{10} & Q_{v1}^{11} & Q_{v2}^{11} & \cdots & Q_{vv}^{11} & R_{v1} \\ \delta_{i1} & \delta_{i2} & \cdots & \delta_{iv} & 0 & 0 & \cdots & 0 & 0 \end{vmatrix}}{\begin{vmatrix} Q_{11}^{00} & \cdots & Q_{1v}^{00} & Q_{11}^{01} & \cdots & Q_{1v}^{01} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Q_{v1}^{00} & \cdots & Q_{vv}^{00} & Q_{v1}^{01} & \cdots & Q_{vv}^{01} \\ Q_{11}^{10} & \cdots & Q_{1v}^{10} & Q_{11}^{11} & \cdots & Q_{1v}^{11} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Q_{v1}^{10} & \cdots & Q_{vv}^{10} & Q_{v1}^{11} & \cdots & Q_{vv}^{11} \end{vmatrix}} \qquad \text{B.41}$$

with

$$R_{im} = \delta_{m1} \frac{15}{4} n_i \sqrt{\frac{2kT}{m_i}} \qquad \text{B.42}$$

Appendix C

C.1 Sonine Polynomials

The Sonine polynomial is defined by

$$S_n^{(m)}(x) = \sum_j \frac{(-1)^j (m+n)!}{(n+j)!(m-j)!\,j!} x^j \qquad \text{C.1}$$

and except for the normalization, these polynomials are the same as the associated Laguerre polynomials. The polynomials satisfy the orthogonality condition

$$\int_0^\infty x^n e^{-x} S_n^{(m)}(x) S_n^{(m')}(x) dx = \frac{(m+n)!}{m!} \delta_{mm'}. \qquad \text{C.2}$$

Two special cases of this orthogonality relation are

$$\int_0^\infty f_i^{[0]} S_{3/2}^{(m)}(W_i^2) V_i^2 dV_i = \frac{3 n_i kT}{m_i} \delta_{m0} \qquad \text{C.3}$$

and

$$\int_0^\infty f_i^{[0]} S_{5/2}^{(m)}(W_i^2) V_i^4 dV_i = 15 n_i \left(\frac{kT}{m_i}\right)^2 \delta_{m0}. \qquad \text{C.4}$$

Defining a finite linear combination of the Sonine polynomial[28,29]

$$t_i^{(h,k)}(W_i) = W_i \sum_{m=0}^{\xi-1} t_{im}^{(h,k)}(\xi) S_n^{(m)}(W_i^2) \qquad \text{C.5}$$

where the meaning of the index n and of the tensor W_i are as follows

When $t_i^{(h,k)}$ is:	The value of the index n is:	The Tensor W_i is	The Sonine expansion coefficient are:
A_i	3/2	W_i	$a_{im}(\xi)$
B_i	5/2	W_iW_i-1/3W_i^2V	$b_{im}(\xi)$
$C_i^{(h)}$- $C_i^{(k)}$	3/2	W_i	$c_{im}(\xi)$

Table C.1

and the equation may be written as

$$\sum_j \sum_{m=0}^{\xi-1} Q_{ij}^{mm'} t_{im'}^{(h,k)}(\xi) = -R_{mi}^{(h,k)} \qquad \text{C.6}$$

where

$$Q_{ij}^{mm'} = \sum_l n_i n_l \begin{bmatrix} \delta_{ij}\left[W_i S_n^{(m)}(W_i^2); W_i S_n^{(m')}(W_i^2)\right]_{il} \\ +\delta_{jl}\left[W_i S_n^{(m)}(W_i^2); W_l S_n^{(m')}(W_{li}^2)\right]_{il} \end{bmatrix} \qquad \text{C.7}$$

The functional form of $\mathbf{A}_i$, $\mathbf{B}_i$, and $\mathbf{C}_i$, defined in the perturbation function defined by B.3, corresponding to a_{jm}, b_{jm}, and c_{jm} respectfully can be summarize as

$$Q_{ij}^{mm'} = \begin{cases} Q_{ij}^{mm'}, & \text{when } t_{im'}^{(h,k)} = b_{jm'} \\ Q_{ij}^{mm'} - \dfrac{n_j\sqrt{m_j}}{n_i\sqrt{m_i}} Q_{ii}^{mm'} \delta_{m0}\delta_{m'0}, & \text{when } t_{im'}^{(h,k)} = a_{jm'} \text{ or } c_{jm'} \end{cases} \qquad \text{C.8}$$

The equation represented by C.7 represents the ξ-1 order approximation of the Sonine Expansion Coefficients designated by Equation C.5. Since we are only interested in the first approximation m and m' are zero and equation C.7 reduces to

$$Q_{ij}^{00} = \sum_l n_i n_l \begin{bmatrix} \delta_{ij}\left[W_i S_n^{(0)}(W_i^2); W_i S_n^{(0)}(W_i^2)\right]_{il} \\ +\delta_{jl}\left[W_i S_n^{(0)}(W_i^2); W_l S_n^{(0)}(W_{li}^2)\right]_{il} \end{bmatrix}$$
$$= \begin{bmatrix} \delta_{ij}\left[\mathbf{W}_i; \mathbf{W}_i\right]_{il} \\ +\delta_{jl}\left[\mathbf{W}_i; \mathbf{W}_l\right]_{il} \end{bmatrix}$$

where Chapman[29] defines

$$[W_i; W_l] = 8\pi^{1/2} M_2^{p+1} M_1^{q+1} \int_0^{\infty}\int_0^{\infty} e^{-\gamma_{ij}^2} \sum A_{pqrl} \gamma_{ij}^{2s+3} \left(1 + \cos^l \chi\right) b\,db\,d\gamma_{ij}$$

where

$$[W_i; W_l] = 8\pi^{1/2} \sum A_{pqrl} \Omega^{(l,s)}$$

where Chapman[29] defines

$$A_{pqrl} = \sum_r \sum_n \left\{ st\left(M_1^2 + M_2^2 + 2M_1 M_2 \cos\chi\right)\right\}^r \left(\gamma^{2r} / r!\right) \left\{M_2(s+t) - (M_2 - M_1)st\right\}^n \left\{M_1(n+1) S_{r+1/2}^{n+1} + (M_1 + M_2 \cos\chi)\gamma^{2r} S_{r+3/2}^n\right\}$$

and

$$\Omega^{(l,s)}(r) = \int_0^{\infty}\int_0^{\infty} e^{-\gamma_{ij}^2} \gamma_{ij}^{2s+3} \left(1 + \cos^l \chi\right) b\,db\,d\gamma_{ij}.$$

Appendix D

Binary Gas Mixtures

D.1 Gamma

Gamma is the ratio of the specific heat at constant pressure (c_p) to the specific heat at constant volume (c_v) and can be expressed as

$$\gamma = \frac{c_p}{c_v}. \qquad \text{D.1}$$

The specific heats for a binary gas mixture can be written as a linear combination of the individual components specific heat

$$c_{p_{12}} = x_1 c_{p_1} + x_2 c_{p_2} \qquad \text{D.2}$$

and

$$c_{v_{12}} = x_1 c_{v_1} + x_2 c_{v_2} \qquad \text{D.3}$$

where x_1 is the mole fraction of species one, x_2 the mole fraction of species two, c_{p1} and c_{v1} are the molar specific heats for species one, and c_{p2} and c_{v2} are the molar specific heats for species two. Gamma for the binary gas mixture can thus be computed by the ratio of Equations (D.2) and (D.3). This is a good approximation for the range of temperatures under consideration.

D.2 Simplified Model

A Pure Gas

The viscosity and thermal conductivity are similar in that they involve the transport of some physical property through the gas. Viscosity is the transport of momentum because of a velocity gradient and thermal conductivity is the transport of thermal energy due to a temperature gradient. The Prandtl number for a binary gas mixture becomes quite complex when the collisions and transfer of momentum and thermal energy that determine the transport coefficients of viscosity and thermal conductivity are considered.

In spite of the complicated behavior of the molecules in a gas a good description of the transport properties of viscosity and thermal conductivity may be obtained if the following assumptions are made[27]:

1. The molecules are rigid, non-attracting spheres with diameter σ and the total scattering cross section is $\sigma_0 = 4\pi\sigma^2$.

2. All molecules travel with the same speed that can be evaluated by integrating B.5 using B.1 as the velocity distribution. The average molecular speed is

$$\bar{v} = \sqrt{\frac{8}{\pi}\frac{kT}{m}}. \qquad \text{D.4}$$

3. All molecules travel in the direction parallel to one of the coordinate axes. One sixth of the molecules travel in the +Z direction, one sixth of the molecules travel in the -Z direction and so forth.

The distance a molecule travels between collisions is defined as the mean free path. In a time interval dt, a molecule moving with speed $\bar{v}$ will travel a distance $\bar{v}\,dt$. The number of collisions suffered by a molecule per unit time in the distance traveled is $\Gamma = \sqrt{2}n\bar{v}\sigma_0$, where n is the mean number of molecules. The mean free path can be expressed as

$$l = \frac{\bar{v}dt}{\Gamma dt} = \frac{\bar{v}}{\sqrt{2}n\bar{v}\sigma_0} = \frac{1}{\sqrt{2}n\sigma_0}. \qquad \text{D.5}$$

This is only true if the molecules are identical otherwise $\sqrt{2}\,\bar{v}$ would have to be replaced with the relative velocity of the two molecules colliding.

Figure D.1 shows three planes separated by a distance equal to the mean free path of the individual molecules located in planes B and A.

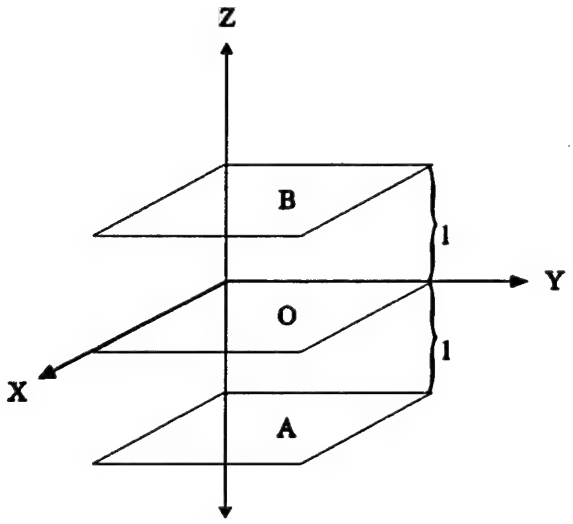

Figure D.1

In planes A and B approximately one-third of the molecules in each plane have velocities in the Z direction. Half of these, 1/6 n have velocities in the - Z direction for plane B and + Z direction for plane A. The net flux of momentum across plane O in the + Z direction is

$$\Psi_p = p_{yz} = -\frac{1}{3}\bar{v}l\frac{dP}{dZ} = -\frac{1}{3}nm\bar{v}l\frac{\partial\bar{v}}{\partial z}. \qquad \text{D.6}$$

The proportionality constant is defined as the viscosity and is written as

$$\eta = \frac{1}{3} nm\bar{v}l = \frac{2}{3}\sqrt{\pi}\frac{\sqrt{mkT}}{\sigma_0}. \qquad \text{D.7}$$

The last equation is written with the use of Equations D.4 and D.5.

The net flux of energy crossing plane O of Figure D.1 is the defined as the heat flow and is

$$\Psi_\varepsilon = q_z = -\frac{1}{6} n\bar{v}l\frac{\partial\bar{\varepsilon}}{\partial z} = -\frac{1}{6} n\bar{v}l\frac{\partial\bar{\varepsilon}}{\partial T}\frac{\partial T}{\partial z}. \qquad \text{D.8}$$

The specific heat at constant volume is defined as $c_v = \partial\bar{\varepsilon}/\partial T$ where $\bar{\varepsilon}$ is the average energy of the molecules. The heat flux becomes

$$q_z = -\frac{1}{6} n\bar{v}lc_v\frac{\partial T}{\partial z}, \qquad \text{D.9}$$

where the thermal conductivity is defined as the proportionality constant and using Equations D.4 and D.5 can be written as

$$\lambda = \frac{2}{3}\frac{c_v}{\sigma_0}\sqrt{\pi}\sqrt{\frac{kT}{m}}. \qquad \text{D.10}$$

The general dependence of viscosity and thermal conductivity on the parameters k, T, m, and σ_0 is correct and shows these transport coefficients are independent of the pressure. The factor of one third is an oversimplification in averaging various quantities. The values for these are calculated in a more rigorous method presented in Appendix B and will be used here. The numerical value for viscosity is 5/16 and for thermal conductivity is 25/32. The viscosity and thermal conductivity can then be written

$$\eta = \frac{5}{16}\sqrt{\pi}\frac{\sqrt{mkT}}{\sigma_0} \qquad \text{D.11}$$

and

$$\lambda = \frac{25}{32}\sqrt{\pi}\frac{c_v}{\sigma_0}\sqrt{\frac{kT}{m}}. \qquad \text{D.12}$$

The Prandtl number is defined as the specific heat at constant pressure times the ratio of the coefficients of viscosity and thermal conductivity. The Prandtl number for a pure gas can be written directly from Equations D.11 and D.12 as

$$N_{pr} = \frac{C_p \eta}{\lambda} = \frac{2}{5}\frac{c_p}{c_v} = \frac{2}{5}\gamma \qquad \text{D.13}$$

For a monatomic gas at 20^0 C Equation D.13 gives $N_{pr} = 0.67$.

B. Simplified Theory for Binary Gas Mixtures

The Prandtl number for binary gas mixtures can be approximated by using Equation D.11, Equation D.12, and the specific heat at constant pressure. The specific heat at constant pressure for a binary monatomic gas mixture is given by Equation D.2 and can be expressed as

$$\left[C_p\right]_{mix} = \frac{R\left(1+\frac{N}{2}\right)}{M_1 x_1 + M_2 x_2} = \frac{R\left(1+\frac{N}{2}\right)}{M_1\left(x_1 + \frac{M_2}{M_1}(1-x_1)\right)} \qquad \text{D.14}$$

where N is the number of degrees of freedom, R is the gas constant, M_1 and M_2 are the molecular weights of species one and two respectively, and x_1 and x_2 are the mole fraction of the species.

The viscosity can be approximated by $[\eta]_{mix} = x_1\eta_1 + x_2\eta_2$ where η_1 is the viscosity of species one and η_2 is the viscosity of species two and x_1 and x_2 are the mole fraction of the species. This can be expanded using Equation D.11 and can be written as

$$[\eta]_{mix} = \frac{5}{16}\sqrt{\pi}\frac{\sqrt{kT}}{\sigma_0}\left(x_1\sqrt{M_1} + x_2\sqrt{M_2}\right)$$
$$= \frac{5}{16}\sqrt{\pi}\frac{\sqrt{kT}}{\sigma_0}\sqrt{M_1}\left(x_1 + \sqrt{\frac{M_2}{M_1}}(1-x_1)\right) \quad \text{D.15}$$

The thermal conductivity for a binary mixture can be written as $[\lambda]_{mix} = \lambda_1/x_1 + \lambda_2/x_2$ and from Equation D.12 can be expanded to

$$[\lambda]_{mix} = \frac{25}{32}\sqrt{\pi}\frac{\sqrt{kT}}{\sigma_0}c_v\left(\frac{1}{x_1\sqrt{M_1}} + \frac{1}{x_2\sqrt{M_2}}\right)$$
$$= \frac{5}{16}\sqrt{\pi}\frac{\sqrt{kT}}{\sigma_0}\frac{1}{\sqrt{m_1}}\left(\frac{1}{x_1} + \frac{1}{x_2\sqrt{\frac{M_1}{M_2}}}\right) \quad \text{D.16}$$

The Prandtl number using Equations D.14, D.15, and D.16 can be written as

$$\left[N_{pr}\right]_{mix} = \frac{2}{5}\left\{\frac{\dfrac{R\left(1+\dfrac{N}{2}\right)}{M_1\left(x_1+\dfrac{M_2}{M_1}(1-x_1)\right)}\sqrt{M_1}\left(x_1+\sqrt{\dfrac{M_2}{M_1}}(1-x_1)\right)}{\dfrac{c_v}{\sqrt{M_1}}\left(\dfrac{1}{x_1}+\dfrac{1}{x_2\sqrt{\dfrac{M_1}{M_2}}}\right)}\right\} \qquad \text{D.17}$$

D.3 Kinetic Theory

A Viscosity

The formulation of the transport property of viscosity for a gas mixture composed of ν components, where ν is the number of component gases, is presented in appendix B.2. The first order approximation for the viscous coefficient for a binary gas mixture can be written as[28]

$$\left[\eta_{mix}\right]_1 = \sum_{i=1}^{2}\frac{x_i^2}{H_{ii}} + \sum_{i=1}^{2}\sum_{\substack{j=1\\ j\neq i}}^{2}\frac{x_i x_j H_{ij}}{H_{ii}H_{jj}}. \qquad \text{D.18}$$

This is a special case of Equation B.22 with H_{ij} defined by Equation B.23 as

$$H_{ij} = \frac{32}{15}\frac{n_i m_i}{n^2 m_j kT}\sum_l \frac{n_l m_l}{(m_i + m_l)^2}\left[\begin{matrix}5m_j(\delta_{ij} - \delta_{jl})\Omega_{il}^{(1,1)} \\ +\frac{3}{2}m_l(\delta_{ij} + \delta_{jl})\Omega_{il}^{(2,2)}\end{matrix}\right]. \qquad \text{D.19}$$

The coefficient of viscosity of a pure gas in the first approximation can be written directly from Equation D.17 with $\nu = 1$ and is

$$[\eta]_1 = \frac{x_1^2}{H_{11}} = \frac{1}{H_{11}} = \frac{5}{8}\frac{kT}{\Omega^{(2,2)}}. \qquad \text{D.20}$$

Where in general[29,30]

$$\Omega_{ij}^{(l,s)} = \sqrt{\frac{2\pi kT}{\mu_{ij}}}\int_0^\infty\int_0^\infty e^{-\gamma_{ij}^2}\gamma_{ij}^{2s+3}\left(1 + \cos^l\chi\right)b\,db\,d\gamma_{ij}. \qquad \text{D.21}$$

Where $\gamma_{ij} = mg^2/2kT$ is the reduced initial relative velocity. These integrals involve explicitly the dynamics of a molecular encounter through the reduced mass, μ_{ij} ,of the colliding molecules, the impact parameter, b, and the angle of deflection,

$$\chi(g,b) = \pi - 2b\int_{r}^{\infty} \frac{dr/r^2}{\sqrt{1-\frac{b^2}{r^2}-\frac{\varphi(r)}{\mu g^2/2}}}. \qquad \text{D.22}$$

φ is the intermolecular interaction potential defined in appendix B and g is the relative reduced velocity. Hirschfelder *et. al.*[28] reduces all quantities by dividing them by their corresponding rigid-sphere values. The collision integral becomes

$$\Omega^{(l,s)}(T) = \frac{\Omega^{(2,2)*}(T^*)}{2} \frac{(s+1)!\left[1-\frac{1+(-1)^l}{2(1+l)}\right]}{\sqrt{\pi m/kT}} \pi\sigma^2, \qquad \text{D.23}$$

this shows the deviation of any particular molecular model from the idealized rigid-sphere model. By Equation D.23

$$\Omega^{(2,2)}(T) = \Omega^{*(2,2)}(T^*) \frac{2\pi\sigma^2}{\sqrt{\pi m/kT}} \qquad \text{D.25}$$

is the reduce collision integral. Substituting into Equation D.20 the viscosity in terms of the reduced quantities is

$$[\eta]_1 = \frac{5}{16}\frac{1}{\sqrt{\pi}}\frac{\sqrt{mkT}}{\sigma^2\Omega^{(2,2)*}(T^*)}. \qquad \text{D.25}$$

Rewriting with $k = 1.38 \cdot 10^{16}$ g cm^2/s and $m=1.66 \cdot 10^{24}$ M, where M is the molecular weight. The coefficient of viscosity for a pure gas is

$$[\eta]_1 \times 10^7 = 266.93\frac{\sqrt{MT}}{\sigma^2\Omega^{(2,2)*}(T^*)} \qquad \text{D.26}$$

where η is the viscosity in g/cm sec, T is the temperature in degrees Kelvin, $T^*=kT/\varepsilon$ is the reduced temperature, M is the molecular weight, σ is the collision diameter is angstroms, ε/k is the potential parameter in degrees Kelvin, and $\Omega^{(2,2)*}(T^*)$ is the collision integral.

Expanding Equation D.18, substituting Equation D.19 appropriately and using Equation D.23 the viscosity of a binary mixture can be written as[28,29,30]

$$\frac{1}{[\eta_{mix}]_1} = \frac{X_\eta + Y_\eta}{1+Z_\eta} = X_\eta\left[\frac{1+Y_\eta/X_\eta}{1+Z_\eta}\right], \qquad \text{D.27}$$

where

$$X_\eta = \frac{x_1^2}{[\eta_1]_1} + \frac{2x_1x_2}{[\eta_{12}]_1} + \frac{x_2^2}{[\eta_2]_1}, \qquad \text{D.28}$$

$$Y_\eta = \frac{3}{5}A_{12}^* \left[\begin{array}{l} \dfrac{x_1^2}{[\eta_1]_1}\left(\dfrac{M_1}{M_2}\right) + \\ \dfrac{2x_1x_2}{[\eta_{12}]_1}\left(\dfrac{(M_1+M_2)^2}{4M_1M_2}\right)\left(\dfrac{[\eta_{12}]_1^2}{[\eta_1]_1[\eta_2]_1}\right) + \dfrac{x_2^2}{[\eta_2]_1}\left(\dfrac{M_1}{M_2}\right) \end{array} \right], \qquad \text{D.29}$$

and

$$Z_\eta = \frac{3}{5}A_{12}^* \left\{ \begin{array}{l} x_1^2\left(\dfrac{M_1}{M_2}\right) + 2x_1x_2\left[\left(\dfrac{(M_1+M_2)^2}{4M_1M_2}\right)\left(\dfrac{[\eta_{12}]_1}{[\eta_1]_1} + \dfrac{[\eta_{12}]_1}{[\eta_2]_1}\right) - 1\right] \\ +x_2^2\left(\dfrac{M_1}{M_2}\right) \end{array} \right\}. \qquad \text{D.30}$$

M_1 and M_2 are the molecular weights of gas 1 and gas 2, x_1 and x_2 are the mole fractions of species 1 and 2. The quantity $A^*_{12} = \Omega^{*(2,2)}/\Omega^{*(1,1)}$ is factored out so we can define a hypothetical viscosity[28] for a gas with molecular weight of $2M_1M_2/(M_1+M_2)$ and collision diameter σ_{12} given by

$$[\eta_{12}]_1 \times 10^7 = 266.93 \frac{\sqrt{2M_1M_2T/(M1+M2)}}{\sigma_{12}^2 \Omega_{12}^{(2,2)*}(T_{12}^*)}, \qquad \text{D.31}$$

where T is the temperature, T_{12} is the reduced temperature, M_1 and M_2 are the molecular weights of species 1 and 2, σ_{12} is the collision parameter

$$\sigma_{12} = \frac{1}{2}(\sigma_1 + \sigma_2), \qquad \text{D.32}$$

and ε_{12}/k is the potential parameter

$$\varepsilon_{12} = \sqrt{\varepsilon_1 \varepsilon_2}\,. \qquad \text{D.33}$$

This is purely a mathematical tool and has no physical significance other than it help streamline Equations D.25-D.30.

The small variance in the viscosity caused by the internal degrees of freedom can be neglected and the viscosity of a polyatomic gas can be adequately described by this method. The viscosity of a binary mixture of monatomic-polyatomic can also be determined using this method.

B. Thermal Conductivity

The thermal conduction for a pure gas is given in the first approximation by Equation B.35 in appendix B when $\nu = 1$ then[28,29,30]

$$[\lambda]_1 \times 10^7 = 1989.1 \frac{\sqrt{T \backslash M}}{\sigma^2 \Omega^{(2,2)^*}(T^*)} = \frac{15}{4}[\eta_1]_1 \times 10^7 . \qquad \text{D.34}$$

The same procedures used to derive Equation D.26 were used here.

The thermal conductivity for a binary mixture of gas can be written using Equations B.35-B.42 as

$$\frac{1}{[\lambda_{mix}]_1} = \frac{X_\lambda + Y_\lambda}{1 + Z_\lambda} = X_\lambda \left[\frac{1 + Y_\lambda / X_\lambda}{1 + Z_\lambda} \right] \qquad \text{D.35}$$

where

$$X_\lambda = \frac{x_1^2}{[\lambda_1]_1} + \frac{2x_1x_2}{[\lambda_{12}]_1} + \frac{x_2^2}{[\lambda_2]_1}, \qquad \text{D.36}$$

$$Y_\lambda = \frac{x_1^2}{[\lambda_1]_1} U^{(1)} + \frac{2x_1x_2}{[\lambda_{12}]_1} U^{(Y)} + \frac{x_2^2}{[\lambda_2]_1} U^{(2)}, \qquad \text{D.37}$$

$$Z_\lambda = x_1^2 U^{(1)} + x_1x_2 U^{(Z)} + x_2^2 U^{(2)}, \qquad \text{D.38}$$

$$U^{(1)} = \frac{4}{15} A_{12}^* - \frac{1}{12}\left(\frac{12}{5} B_{12}^* + 1\right)\frac{M_1}{M_2} + \frac{1}{2}\frac{(M_1 - M_2)^2}{M_1M_2}, \qquad \text{D.39}$$

$$U^{(2)} = \frac{4}{15} A_{12}^* - \frac{1}{12}\left(\frac{12}{5} B_{12}^* + 1\right)\frac{M_2}{M_1} + \frac{1}{2}\frac{(M_2 - M_1)^2}{M_1M_2}, \qquad \text{D.40}$$

$$U^{(Y)} = \frac{4}{15} A_{12}^* \left(\frac{(M_1 + M_2)^2}{4M_1M_2}\right)\frac{[\lambda_{12}]_1}{[\lambda_1]_1[\lambda_2]_2} - \frac{1}{12}\left(\frac{12}{5} B_{12}^* + 1\right)$$
$$- \frac{5}{32A_{12}^*}\frac{1}{12}\left(\frac{12}{5} B_{12}^* - 5\right)\frac{(M_1 - M_2)^2}{M_1M_2}, \qquad \text{D.41}$$

and

$$U^{(Z)} = \frac{4}{15} A_{12}^* \left[\left(\frac{(M_1 + M_2)^2}{4M_1M_2}\right)\left(\frac{[\lambda_{12}]_1}{[\lambda_1]_1} + \frac{[\lambda_{12}]_1}{[\lambda_2]_1}\right) - 1\right] - \frac{1}{12}\left(\frac{12}{5} B_{12}^* + 1\right). \qquad \text{D.42}$$

The terms $A^*_{12} = \Omega^{*(2,2)}/\Omega^{*(1,1)}$ and $B^*_{12} = (5\Omega^{*(1,2)} - 4\Omega^{*(1,3)})/\Omega^{*(1,1)}$ have no significance other than they are ratio of collision integrals and help reduce the length of the above equation by defining the thermal conductivity of the hypothetical species as

$$[\lambda_{12}]_1 \times 10^7 = 1989.1 \frac{\sqrt{T(M_1 + M_2)/2M_1M_2}}{\sigma_{12}^2 \Omega_{12}^{(2,2)*}(T^*)}. \qquad \text{D.44}$$

where the variables are the same as defined in the hypothetical viscosity. For polyatomic molecules the thermal conductivity should be modified with the Eucken[30] approximation to take into consideration the internal energy;

$$[\lambda]_1 \times 10^7 = = \frac{15}{4}\frac{R}{M}[\eta_1]_1\left(\frac{4}{15}\frac{C_v}{R} + \frac{3}{5}\right) \times 10^7. \qquad \text{D.45}$$

For monatomic gases C_v=3R/2 and the Eucken equation reduces to Equation D.34.

A better approximation to the thermal conductivity for binary mixtures of monatomic-polyatomic gas is to calculate the mixture assuming it is monatomic. If experimental values are known, we can define $E_i = \lambda_{iexp}/\lambda_{imon}$ and the empirical expression for the mixture can be written as[28,30]

$$\lambda_{mix} = \lambda_{mon}(x_1E_1 + x_2E_2). \qquad \text{D.46}$$

Appendix E

The binary gas mixture program described in Chapter 4 and Appendix B.

Program Mixtures

```
**************************************************************************
* This program uses the method outlined in Chapter 4 and Appendix D      *
**************************************************************************

**************************************************************************
Variable declarations.
implicit none
integer column,kind
Character*20 fileout
double precision s_1, s_2, s_1_2, ek_1, ek_2, ek_1_2, viscosity
double precision prandtl, A, B, X_therm, Y_therm, Z_therm, X_visc,i
double precision Y_visc, Z_visc, frac_1, frac_2, T, Tstar_1
double precision Tstar_1_2, Omega_1, Omega_2, Omega_1_2, M_1, M_2
double precision omega(3,82), alpha(3,82), eta_1, eta_2, eta_1_2, C_P
double precision lam_1, lam_2, lam_1_2, therm_cond, U_1, U_2, U_Y
double precision U_Z, Tstar_2, f(3,82), f_visc_1, f_visc_2, f_visc_1_2
double precision f_therm_1, f_therm_2, f_therm_1_2,e1,e2,lam_11,lam_22
double precision pi,cv,gamma
double precision cp1,cp2,cv1,cv2

**************************************************************************
print*,'Enter the name of the output file = '
read*,fileout
open (10, fileout)
10 format (1x,1f5.3,5f20.8)
print*,'Enter the Temperature',T

**************************************************************************

*NECESSARY INPUTS FROM TABLE 4.1
print*,'NOTE THE MONATOMIC GAS MUST BE ENTERED FIRST'
print*,'NOTE THE MONATOMIC GAS MUST BE ENTERED FIRST'
print*,'Enter the molecular weight of gas 1 (grams)= '
```

```
read*,M_1
print*,'Enter the Cp (cal/mole k) of gas 1 = '
read*,cp1
print*,'Enter the Cv (cal/moleK) of gas 1  = '
read*,cv1
print*,'Enter the cross section of gas 1 (Angs) = '
read*,s_1
print*,'Enter the eps/k (degrees k) of gas 1  = '
read*,ek_1
print*,'Enter the molecular weight of gas 2 = '
read*,M_2
print*,'Enter the Cp (cal/mol K) of gas 2 = '
read*,cp2
print*,'Enter the Cv (cal/mol K) of gas 2 = '
read*,cv2
print*,'Enter the cross section of gas 2 (angs) = '
read*,s_2
print*,'Enter the eps/k (degreesK) of gas 2 = '
read*,ek_2
print*,'For binary gas mixtures of monatomic gases please enter 1. '
print*,'For binary gas mixtures of monatomic-polyatomic enter 2. '
print*,'For all other gas mixtures please enter 3. '
read*,kind
if (kind .gt. 1) then
print*,'For mixtures other than monatomic you must'
print*,'enter the experimental values of thermal'
print*,'conductives in units of (cal/(cm*sec*K))'
print*,'Enter the value for gas 1 '
read*,lam_11
print*,'Enter the value for gas 2 '
read*,lam_22
else
endif

DO i=0.D0,1.0D0,0.01D0
  FRAC_1 = i
  FRAC_2 = 1.0D0 - FRAC_1
 C_P = frac_1 * cp1 + frac_2 * cp2
Cv = (cv2*frac_2) + (cv1*frac_1)
```

```
************************************************************************
************************************************************************

* Scattering cross-sections (s_???) and potential parameters (ek_???)
* are found in table I-A.  The units are Angstroms and degrees Kelvin
* respectively.

s_1_2 = (s_1 + s_2) / 2.d0
ek_1_2 = DSQRT(ek_1 * ek_2)

************************************************************************
************************************************************************

* Tstar is called the reduced temperature.  It is the ambient
* temperature divided by (ek_???).  It is a unitless quantity.
Tstar_1 = T / ek_1
Tstar_2 = T / ek_2
Tstar_1_2 = T / ek_1_2

************************************************************************
************************************************************************

* The following two statements fill "omega" and "alpha" with the
* relevant values from tables I-M and I-N.  See example on p. 569
* for further information.
Call Omega_def(omega)
Call Alpha_def(alpha)
Call f_def(f)

************************************************************************
************************************************************************

* The following statements choose the correct values of "omega", "A",
* "B", and "f" from tables I-M, I-N, and I-P.
 column = 2
 Call Chooser(omega, Omega_1, Tstar_1, column)
 Call Chooser(omega, Omega_2, Tstar_2, column)
 Call Chooser(omega, Omega_1_2, Tstar_1_2, column)
 Call Chooser(alpha, A, Tstar_1_2, column)
 Call Chooser(f, f_visc_1, Tstar_1, column)
```

```
 Call Chooser(f, f_visc_2, Tstar_2, column)
 Call Chooser(f, f_visc_1_2, Tstar_1_2, column)
 column = 3
 Call Chooser(alpha, B, Tstar_1_2, column)
 Call Chooser(f, f_therm_1, Tstar_1, column)
 Call Chooser(f, f_therm_2, Tstar_2, column)
 Call Chooser(f, f_therm_1_2, Tstar_1_2, column)

************************************************************************
************************************************************************
* Solve for the coefficient of viscosity.  See pp. 528-530. The units
* are ... grams/(cm * sec).
 eta_1 = 266.93d-7 * DSQRT(M_1 * T) / (s_1**2 * Omega_1)
 eta_1 = eta_1 * f_visc_1
 eta_2 = 266.93d-7 * DSQRT(M_2 * T) / (s_2**2 * Omega_2)
 eta_2 = eta_2 * f_visc_2
 eta_1_2 = 266.93d-7 * DSQRT(2.d0 * M_1 * M_2 * T / (M_1 + M_2)) /
&          (s_1_2**2 * Omega_1_2)
 eta_1_2 = eta_1_2 * f_visc_1_2

 X_visc = (frac_1**2 / eta_1)
 X_visc = X_visc + (frac_2**2 / eta_2)
 X_visc = X_visc + (2.d0 * frac_1 * frac_2 / eta_1_2)

 Y_visc = (frac_1**2 / eta_1)*(M_1 / M_2)
 Y_visc = Y_visc + (2.d0 * frac_1 * frac_2 / eta_1_2)*((M_1 + M_2)**2
&          / (4.d0 * M_1 * M_2))*(eta_1_2**2 / (eta_1 * eta_2))
 Y_visc = Y_visc + (frac_2**2 / eta_2)*(M_2 / M_1)
 Y_visc = 3.d0 * A * Y_visc / 5.d0

 Z_visc = eta_1_2 / eta_1 + eta_1_2 / eta_2
 Z_visc = Z_visc * (M_1 + M_2)**2 / (4.d0 * M_1 * M_2)
 Z_visc = Z_visc - 1.d0
 Z_visc = 2.d0 * frac_1 * frac_2 * Z_visc
 Z_visc = Z_visc + frac_1**2 * M_1 / M_2
 Z_visc = Z_visc + frac_2**2 * M_2 / M_1
 Z_visc = Z_visc * 3.d0 * A / 5.d0

 viscosity = (X_visc + Y_visc) / (1.d0 + Z_visc)
 viscosity = 1.d0 / viscosity
```

```
************************************************************************
************************************************************************

* Solve for the coefficient of thermal conductivity.  See pp. 533-535
* for reference.  The units are ... cal/(cm * sec * K)

 if( kind.eq.2) then

 lam_1 = 1989.1d-7 * dsqrt(T/M_1)/(s_1**2 * Omega_1)
 lam_1 = lam_1*f_therm_2

 lam_2 = 1989.1d-7 * DSQRT(T / M_2) / (s_2**2 * Omega_2)
 lam_2 = lam_2*(.1346d0*cv2+.6d0)* f_therm_2

 e1 = lam_11/lam_1
 e2 = lam_22/lam_2

 lam_1_2 = 1989.1d-7 * DSQRT(T*(M_1 + M_2) / (2.d0 * M_1 * M_2)) /
&            (s_1_2**2 * Omega_1_2)
 lam_1_2 = lam_1_2*(.1346d0*cv+.6d0) * f_therm_1_2

 U_1 = (4.d0 * A / 15.d0)
 U_1 = U_1 - (1.d0 / 12.d0) * (12.d0 * B / 5.d0 + 1.d0) * (M_1 / M_2)
 U_1 = U_1 + (M_1 - M_2)**2 / (2.d0 * M_1 * M_2)

 U_2 = (4.d0 * A / 15.d0)
 U_2 = U_2 - (1.d0 / 12.d0) * (12.d0 * B / 5.d0 + 1.d0) * (M_2 / M_1)
 U_2 = U_2 + (M_2 - M_1)**2 / (2.d0 * M_1 * M_2)

 U_Y = (4.d0 * A / 15.d0) * ((M_1 + M_2)**2 / (4.d0 * M_1 * M_2)) *
&       (lam_1_2**2 / (lam_1 * lam_2))
 U_Y = U_Y - (1.d0 / 12.d0) * (12.d0 * B / 5.d0 + 1.d0)
 U_Y = U_Y - (5.d0 / (32.d0 * A)) * (12.d0 * B / 5.d0 - 5.d0)
&       * ((M_1 - M_2)**2 / (M_1 * M_2))

 U_Z = (M_1 + M_2)**2 / (4.d0 * M_1 * M_2)
 U_Z = U_Z * ((lam_1_2 / lam_1) + (lam_1_2 / lam_2))
 U_Z = U_Z - 1.d0
 U_Z = U_Z * 4.d0 * A / 15.d0
```

```
 U_Z = U_Z - (1.d0 / 12.d0)*(12.d0 * B / 5.d0 + 1.d0)

 X_therm = (frac_1**2 / lam_1) + (2.d0 * frac_1 * frac_2 /
&          lam_1_2) + (frac_2**2 / lam_2)

 Y_therm = (frac_1**2 * U_1 / lam_1) + (2.d0 * frac_1 * frac_2 *
&          U_Y / lam_1_2) + (frac_2**2 * U_2 / lam_2)

 Z_therm = (frac_1**2 * U_1) + (2.d0 * frac_1 * frac_2 * U_Z)
&          + (frac_2**2 * U_2)

 therm_cond = (X_therm + Y_therm) / ( 1 + Z_therm)
 therm_cond = 1.d0 / therm_cond
 therm_cond = therm_cond*(frac_1 * e1 + frac_2 * e2)

else IF (kind .eq. 1) then

 lam_1 = 1989.1d-7 * dsqrt(T/M_1)/(s_1**2 * Omega_1)
 lam_1 = lam_1*f_therm_2

 lam_2 = 1989.1d-7 * DSQRT(T / M_2) / (s_2**2 * Omega_2)
 lam_2 = lam_2* f_therm_2

 lam_1_2 = 1989.1d-7 * DSQRT(T*(M_1 + M_2) / (2.d0 * M_1 * M_2)) /
&          (s_1_2**2 * Omega_1_2)

 U_1 = (4.d0 * A / 15.d0)
 U_1 = U_1 - (1.d0 / 12.d0) * (12.d0 * B / 5.d0 + 1.d0) * (M_1 / M_2)
 U_1 = U_1 + (M_1 - M_2)**2 / (2.d0 * M_1 * M_2)

 U_2 = (4.d0 * A / 15.d0)
 U_2 = U_2 - (1.d0 / 12.d0) * (12.d0 * B / 5.d0 + 1.d0) * (M_2 / M_1)
 U_2 = U_2 + (M_2 - M_1)**2 / (2.d0 * M_1 * M_2)

 U_Y = (4.d0 * A / 15.d0) * ((M_1 + M_2)**2 / (4.d0 * M_1 * M_2)) *
&       (lam_1_2**2 / (lam_1 * lam_2))
 U_Y = U_Y - (1.d0 / 12.d0) * (12.d0 * B / 5.d0 + 1.d0)
 U_Y = U_Y - (5.d0 / (32.d0 * A)) * (12.d0 * B / 5.d0 - 5.d0)
&       * ((M_1 - M_2)**2 / (M_1 * M_2))
```

```
      U_Z = (M_1 + M_2)**2 / (4.d0 * M_1 * M_2)
      U_Z = U_Z * ((lam_1_2 / lam_1) + (lam_1_2 / lam_2))
      U_Z = U_Z - 1.d0
      U_Z = U_Z * 4.d0 * A / 15.d0
      U_Z = U_Z - (1.d0 / 12.d0)*(12.d0 * B / 5.d0 + 1.d0)

      X_therm = (frac_1**2 / lam_1) + (2.d0 * frac_1 * frac_2 /
     &          lam_1_2) + (frac_2**2 / lam_2)

      Y_therm = (frac_1**2 * U_1 / lam_1) + (2.d0 * frac_1 * frac_2 *
     &          U_Y / lam_1_2) + (frac_2**2 * U_2 / lam_2)

      Z_therm = (frac_1**2 * U_1) + (2.d0 * frac_1 * frac_2 * U_Z)
     &          + (frac_2**2 * U_2)

      therm_cond = (X_therm + Y_therm) / ( 1 + Z_therm)
      therm_cond = 1.d0 / therm_cond
    else

      lam_1 = 1989.1d-7 * dsqrt(T/M_1)/(s_1**2 * Omega_1)
      lam_1 = lam_1*(.1346d0*cv1+.6d0)*f_therm_2

      lam_2 = 1989.1d-7 * DSQRT(T / M_2) / (s_2**2 * Omega_2)
      lam_2 = lam_2*(.1346d0*cv2+.6d0)* f_therm_2

      e1 = lam_11/lam_1
      e2 = lam_22/lam_2

      lam_1_2 = 1989.1d-7 * DSQRT(T*(M_1 + M_2) / (2.d0 * M_1 * M_2)) /
     &          (s_1_2**2 * Omega_1_2)
      lam_1_2 = lam_1_2*(.1346d0*cv+.6d0) * f_therm_1_2

      U_1 = (4.d0 * A / 15.d0)
      U_1 = U_1 - (1.d0 / 12.d0) * (12.d0 * B / 5.d0 + 1.d0) * (M_1 / M_2)
      U_1 = U_1 + (M_1 - M_2)**2 / (2.d0 * M_1 * M_2)

      U_2 = (4.d0 * A / 15.d0)
      U_2 = U_2 - (1.d0 / 12.d0) * (12.d0 * B / 5.d0 + 1.d0) * (M_2 / M_1)
      U_2 = U_2 + (M_2 - M_1)**2 / (2.d0 * M_1 * M_2)
```

```
 U_Y = (4.d0 * A / 15.d0) * ((M_1 + M_2)**2 / (4.d0 * M_1 * M_2)) *
&        (lam_1_2**2 / (lam_1 * lam_2))
 U_Y = U_Y - (1.d0 / 12.d0) * (12.d0 * B / 5.d0 + 1.d0)
 U_Y = U_Y - (5.d0 / (32.d0 * A)) * (12.d0 * B / 5.d0 - 5.d0)
&        * ((M_1 - M_2)**2 / (M_1 * M_2))

 U_Z = (M_1 + M_2)**2 / (4.d0 * M_1 * M_2)
 U_Z = U_Z * ((lam_1_2 / lam_1) + (lam_1_2 / lam_2))
 U_Z = U_Z - 1.d0
 U_Z = U_Z * 4.d0 * A / 15.d0
 U_Z = U_Z - (1.d0 / 12.d0)*(12.d0 * B / 5.d0 + 1.d0)

 X_therm = (frac_1**2 / lam_1) + (2.d0 * frac_1 * frac_2 /
&            lam_1_2) + (frac_2**2 / lam_2)

 Y_therm = (frac_1**2 * U_1 / lam_1) + (2.d0 * frac_1 * frac_2 *
&            U_Y / lam_1_2) + (frac_2**2 * U_2 / lam_2)

 Z_therm = (frac_1**2 * U_1) + (2.d0 * frac_1 * frac_2 * U_Z)
&            + (frac_2**2 * U_2)

 therm_cond = (X_therm + Y_therm) / ( 1 + Z_therm)
 therm_cond = 1.d0 / therm_cond
 therm_cond = therm_cond*(frac_1 * e1 + frac_2 * e2)

 endif

**************************************************************************
**************************************************************************

* Prandtl number calculation.  Prandtl number is unitless.
Prandtl =(C_P/(frac_1*M_1+frac_2*M_2))*viscosity / therm_cond
**************************************************************************
**************************************************************************

 c_p= C_P/(frac_1*M_1+frac_2*M_2)
 cv= Cv/(frac_1*M_1+frac_2*M_2)
 gamma = c_p/cv
 write(10,10) prandtl,c_p,viscosity,therm_cond,gamma,frac_1
end do
```

```
end

subroutine Omega_def(dummy)
************************************************************************
* This subprogram defines the values of Omega from table I_M for       *
* different values of Tstar                                            *
************************************************************************

implicit none

double precision dummy(3,82)

dummy(1,1) = 0.3d0
dummy(1,2) = 0.35d0
dummy(1,3) = 0.4d0
dummy(1,4) = 0.45d0
dummy(1,5) = 0.5d0
dummy(1,6) = 0.55d0
dummy(1,7) = 0.6d0
dummy(1,8) = 0.65d0
dummy(1,9) = 0.7d0
dummy(1,10) = 0.75d0
dummy(1,11) = 0.8d0
dummy(1,12) = 0.85d0
dummy(1,13) = 0.9d0
dummy(1,14) = 0.95d0
dummy(1,15) = 1.0d0
dummy(1,16) = 1.05d0
dummy(1,17) = 1.1d0
dummy(1,18) = 1.15d0
dummy(1,19) = 1.2d0
dummy(1,20) = 1.25d0
dummy(1,21) = 1.3d0
dummy(1,22) = 1.35d0
dummy(1,23) = 1.4d0
dummy(1,24) = 1.45d0
dummy(1,25) = 1.5d0
dummy(1,26) = 1.55d0
dummy(1,27) = 1.6d0
dummy(1,28) = 1.65d0
```

```
dummy(1,29) = 1.7d0
dummy(1,30) = 1.75d0
dummy(1,31) = 1.8d0
dummy(1,32) = 1.85d0
dummy(1,33) = 1.9d0
dummy(1,34) = 1.95d0
dummy(1,35) = 2.0d0
dummy(1,36) = 2.1d0
dummy(1,37) = 2.2d0
dummy(1,38) = 2.3d0
dummy(1,39) = 2.4d0
dummy(1,40) = 2.5d0
dummy(1,41) = 2.6d0
dummy(1,42) = 2.7d0
dummy(1,43) = 2.8d0
dummy(1,44) = 2.9d0
dummy(1,45) = 3.0d0
dummy(1,46) = 3.1d0
dummy(1,47) = 3.2d0
dummy(1,48) = 3.3d0
dummy(1,49) = 3.4d0
dummy(1,50) = 3.5d0
dummy(1,51) = 3.6d0
dummy(1,52) = 3.7d0
dummy(1,53) = 3.8d0
dummy(1,54) = 3.9d0
dummy(1,55) = 4.0d0
dummy(1,56) = 4.1d0
dummy(1,57) = 4.2d0
dummy(1,58) = 4.3d0
dummy(1,59) = 4.4d0
dummy(1,60) = 4.5d0
dummy(1,61) = 4.6d0
dummy(1,62) = 4.7d0
dummy(1,63) = 4.8d0
dummy(1,64) = 4.9d0
dummy(1,65) = 5.0d0
dummy(1,66) = 6.0d0
dummy(1,67) = 7.0d0
dummy(1,68) = 8.0d0
```

```
dummy(1,69) = 9.0d0
dummy(1,70) = 10.0d0
dummy(1,71) = 20.0d0
dummy(1,72) = 30.0d0
dummy(1,73) = 40.0d0
dummy(1,74) = 50.0d0
dummy(1,75) = 60.0d0
dummy(1,76) = 70.0d0
dummy(1,77) = 80.0d0
dummy(1,78) = 90.0d0
dummy(1,79) = 100.0d0
dummy(1,80) = 200.0d0
dummy(1,81) = 300.0d0
dummy(1,82) = 400.0d0

dummy(2,1) = 2.785d0
dummy(2,2) = 2.628d0
dummy(2,3) = 2.492d0
dummy(2,4) = 2.368d0
dummy(2,5) = 2.257d0
dummy(2,6) = 2.156d0
dummy(2,7) = 2.065d0
dummy(2,8) = 1.982d0
dummy(2,9) = 1.908d0
dummy(2,10) = 1.841d0
dummy(2,11) = 1.78d0
dummy(2,12) = 1.725d0
dummy(2,13) = 1.675d0
dummy(2,14) = 1.629d0
dummy(2,15) = 1.587d0
dummy(2,16) = 1.549d0
dummy(2,17) = 1.514d0
dummy(2,18) = 1.482d0
dummy(2,19) = 1.452d0
dummy(2,20) = 1.424d0
dummy(2,21) = 1.399d0
dummy(2,22) = 1.375d0
dummy(2,23) = 1.353d0
dummy(2,24) = 1.333d0
dummy(2,25) = 1.314d0
```

```
dummy(2,26) = 1.296d0
dummy(2,27) = 1.279d0
dummy(2,28) = 1.264d0
dummy(2,29) = 1.248d0
dummy(2,30) = 1.234d0
dummy(2,31) = 1.221d0
dummy(2,32) = 1.209d0
dummy(2,33) = 1.197d0
dummy(2,34) = 1.186d0
dummy(2,35) = 1.175d0
dummy(2,36) = 1.156d0
dummy(2,37) = 1.138d0
dummy(2,38) = 1.122d0
dummy(2,39) = 1.107d0
dummy(2,40) = 1.093d0
dummy(2,41) = 1.081d0
dummy(2,42) = 1.069d0
dummy(2,43) = 1.058d0
dummy(2,44) = 1.048d0
dummy(2,45) = 1.039d0
dummy(2,46) = 1.03d0
dummy(2,47) = 1.022d0
dummy(2,48) = 1.014d0
dummy(2,49) = 1.007d0
dummy(2,50) = 0.9999d0
dummy(2,51) = 0.9932d0
dummy(2,52) = 0.9870d0
dummy(2,53) = 0.9811d0
dummy(2,54) = 0.9755d0
dummy(2,55) = 0.97d0
dummy(2,56) = 0.9649d0
dummy(2,57) = 0.96d0
dummy(2,58) = 0.9553d0
dummy(2,59) = 0.9507d0
dummy(2,60) = 0.9464d0
dummy(2,61) = 0.9422d0
dummy(2,62) = 0.9382d0
dummy(2,63) = 0.9343d0
dummy(2,64) = 0.9305d0
dummy(2,65) = 0.9269d0
```

```
dummy(2,66) = 0.8963d0
dummy(2,67) = 0.8727d0
dummy(2,68) = 0.8538d0
dummy(2,69) = 0.8379d0
dummy(2,70) = 0.8242d0
dummy(2,71) = 0.7432d0
dummy(2,72) = 0.7005d0
dummy(2,73) = 0.6718d0
dummy(2,74) = 0.6504d0
dummy(2,75) = 0.6335d0
dummy(2,76) = 0.6194d0
dummy(2,77) = 0.6076d0
dummy(2,78) = 0.5973d0
dummy(2,79) = 0.5882d0
dummy(2,80) = 0.5320d0
dummy(2,81) = 0.5016d0
dummy(2,82) = 0.4811d0

return
end
*************************************************************************
subroutine Alpha_def(dummy)
*************************************************************************
* This subprogram defines the values of A and B from table I_N for      *
* different values of Tstar.                                            *
*************************************************************************

implicit none

double precision dummy(3,82)
dummy(1,1) = 0.3d0
dummy(1,2) = 0.35d0
dummy(1,3) = 0.4d0
dummy(1,4) = 0.45d0
dummy(1,5) = 0.5d0
dummy(1,6) = 0.55d0
dummy(1,7) = 0.6d0
dummy(1,8) = 0.65d0
dummy(1,9) = 0.7d0
dummy(1,10) = 0.75d0
```

```
dummy(1,11) = 0.8d0
dummy(1,12) = 0.85d0
dummy(1,13) = 0.9d0
dummy(1,14) = 0.95d0
dummy(1,15) = 1.0d0
dummy(1,16) = 1.05d0
dummy(1,17) = 1.1d0
dummy(1,18) = 1.15d0
dummy(1,19) = 1.2d0
dummy(1,20) = 1.25d0
dummy(1,21) = 1.3d0
dummy(1,22) = 1.35d0
dummy(1,23) = 1.4d0
dummy(1,24) = 1.45d0
dummy(1,25) = 1.5d0
dummy(1,26) = 1.55d0
dummy(1,27) = 1.6d0
dummy(1,28) = 1.65d0
dummy(1,29) = 1.7d0
dummy(1,30) = 1.75d0
dummy(1,31) = 1.8d0
dummy(1,32) = 1.85d0
dummy(1,33) = 1.9d0
dummy(1,34) = 1.95d0
dummy(1,35) = 2.0d0
dummy(1,36) = 2.1d0
dummy(1,37) = 2.2d0
dummy(1,38) = 2.3d0
dummy(1,39) = 2.4d0
dummy(1,40) = 2.5d0
dummy(1,41) = 2.6d0
dummy(1,42) = 2.7d0
dummy(1,43) = 2.8d0
dummy(1,44) = 2.9d0
dummy(1,45) = 3.0d0
dummy(1,46) = 3.1d0
dummy(1,47) = 3.2d0
dummy(1,48) = 3.3d0
dummy(1,49) = 3.4d0
dummy(1,50) = 3.5d0
```

```
dummy(1,51) = 3.6d0
dummy(1,52) = 3.7d0
dummy(1,53) = 3.8d0
dummy(1,54) = 3.9d0
dummy(1,55) = 4.0d0
dummy(1,56) = 4.1d0
dummy(1,57) = 4.2d0
dummy(1,58) = 4.3d0
dummy(1,59) = 4.4d0
dummy(1,60) = 4.5d0
dummy(1,61) = 4.6d0
dummy(1,62) = 4.7d0
dummy(1,63) = 4.8d0
dummy(1,64) = 4.9d0
dummy(1,65) = 5.0d0
dummy(1,66) = 6.0d0
dummy(1,67) = 7.0d0
dummy(1,68) = 8.0d0
dummy(1,69) = 9.0d0
dummy(1,70) = 10.0d0
dummy(1,71) = 20.0d0
dummy(1,72) = 30.0d0
dummy(1,73) = 40.0d0
dummy(1,74) = 50.0d0
dummy(1,75) = 60.0d0
dummy(1,76) = 70.0d0
dummy(1,77) = 80.0d0
dummy(1,78) = 90.0d0
dummy(1,79) = 100.0d0
dummy(1,80) = 200.0d0
dummy(1,81) = 300.0d0
dummy(1,82) = 400.0d0

dummy(2,1) = 1.046d0
dummy(2,2) = 1.062d0
dummy(2,3) = 1.075d0
dummy(2,4) = 1.084d0
dummy(2,5) = 1.093d0
dummy(2,6) = 1.097d0
dummy(2,7) = 1.101d0
```

```
dummy(2,8) = 1.102d0
dummy(2,9) = 1.104d0
dummy(2,10) = 1.105d0
dummy(2,11) = 1.105d0
dummy(2,12) = 1.105d0
dummy(2,13) = 1.104d0
dummy(2,14) = 1.103d0
dummy(2,15) = 1.103d0
dummy(2,16) = 1.102d0
dummy(2,17) = 1.102d0
dummy(2,18) = 1.101d0
dummy(2,19) = 1.100d0
dummy(2,20) = 1.099d0
dummy(2,21) = 1.099d0
dummy(2,22) = 1.098d0
dummy(2,23) = 1.097d0
dummy(2,24) = 1.097d0
dummy(2,25) = 1.097d0
dummy(2,26) = 1.096d0
dummy(2,27) = 1.096d0
dummy(2,28) = 1.096d0
dummy(2,29) = 1.095d0
dummy(2,30) = 1.094d0
dummy(2,31) = 1.094d0
dummy(2,32) = 1.094d0
dummy(2,33) = 1.094d0
dummy(2,34) = 1.094d0
dummy(2,35) = 1.094d0
dummy(2,36) = 1.094d0
dummy(2,37) = 1.094d0
dummy(2,38) = 1.094d0
dummy(2,39) = 1.094d0
dummy(2,40) = 1.094d0
dummy(2,41) = 1.094d0
dummy(2,42) = 1.094d0
dummy(2,43) = 1.094d0
dummy(2,44) = 1.095d0
dummy(2,45) = 1.095d0
dummy(2,46) = 1.095d0
dummy(2,47) = 1.096d0
```

dummy(2,48) = 1.096d0
dummy(2,49) = 1.096d0
dummy(2,50) = 1.097d0
dummy(2,51) = 1.097d0
dummy(2,52) = 1.097d0
dummy(2,53) = 1.097d0
dummy(2,54) = 1.097d0
dummy(2,55) = 1.098d0
dummy(2,56) = 1.098d0
dummy(2,57) = 1.098d0
dummy(2,58) = 1.099d0
dummy(2,59) = 1.099d0
dummy(2,60) = 1.099d0
dummy(2,61) = 1.100d0
dummy(2,62) = 1.100d0
dummy(2,63) = 1.100d0
dummy(2,64) = 1.101d0
dummy(2,65) = 1.101d0
dummy(2,66) = 1.103d0
dummy(2,67) = 1.105d0
dummy(2,68) = 1.107d0
dummy(2,69) = 1.109d0
dummy(2,70) = 1.110d0
dummy(2,71) = 1.119d0
dummy(2,72) = 1.124d0
dummy(2,73) = 1.127d0
dummy(2,74) = 1.130d0
dummy(2,75) = 1.132d0
dummy(2,76) = 1.134d0
dummy(2,77) = 1.135d0
dummy(2,78) = 1.137d0
dummy(2,79) = 1.138d0
dummy(2,80) = 1.146d0
dummy(2,81) = 1.151d0
dummy(2,82) = 1.154d0

dummy(3,1) = 1.289d0
dummy(3,2) = 1.296d0
dummy(3,3) = 1.296d0
dummy(3,4) = 1.289d0

```
dummy(3,5) = 1.284d0
dummy(3,6) = 1.275d0
dummy(3,7) = 1.263d0
dummy(3,8) = 1.254d0
dummy(3,9) = 1.242d0
dummy(3,10) = 1.233d0
dummy(3,11) = 1.223d0
dummy(3,12) = 1.216d0
dummy(3,13) = 1.206d0
dummy(3,14) = 1.200d0
dummy(3,15) = 1.192d0
dummy(3,16) = 1.183d0
dummy(3,17) = 1.179d0
dummy(3,18) = 1.172d0
dummy(3,19) = 1.169d0
dummy(3,20) = 1.165d0
dummy(3,21) = 1.159d0
dummy(3,22) = 1.156d0
dummy(3,23) = 1.154d0
dummy(3,24) = 1.148d0
dummy(3,25) = 1.143d0
dummy(3,26) = 1.140d0
dummy(3,27) = 1.137d0
dummy(3,28) = 1.137d0
dummy(3,29) = 1.133d0
dummy(3,30) = 1.129d0
dummy(3,31) = 1.127d0
dummy(3,32) = 1.126d0
dummy(3,33) = 1.124d0
dummy(3,34) = 1.122d0
dummy(3,35) = 1.119d0
dummy(3,36) = 1.116d0
dummy(3,37) = 1.113d0
dummy(3,38) = 1.110d0
dummy(3,39) = 1.108d0
dummy(3,40) = 1.106d0
dummy(3,41) = 1.104d0
dummy(3,42) = 1.103d0
dummy(3,43) = 1.104d0
dummy(3,44) = 1.102d0
```

```
dummy(3,45) = 1.101d0
dummy(3,46) = 1.100d0
dummy(3,47) = 1.096d0
dummy(3,48) = 1.099d0
dummy(3,49) = 1.096d0
dummy(3,50) = 1.096d0
dummy(3,51) = 1.095d0
dummy(3,52) = 1.097d0
dummy(3,53) = 1.093d0
dummy(3,54) = 1.093d0
dummy(3,55) = 1.095d0
dummy(3,56) = 1.093d0
dummy(3,57) = 1.093d0
dummy(3,58) = 1.094d0
dummy(3,59) = 1.094d0
dummy(3,60) = 1.092d0
dummy(3,61) = 1.091d0
dummy(3,62) = 1.093d0
dummy(3,63) = 1.095d0
dummy(3,64) = 1.091d0
dummy(3,65) = 1.092d0
dummy(3,66) = 1.090d0
dummy(3,67) = 1.092d0
dummy(3,68) = 1.090d0
dummy(3,69) = 1.091d0
dummy(3,70) = 1.094d0
dummy(3,71) = 1.095d0
dummy(3,72) = 1.095d0
dummy(3,73) = 1.095d0
dummy(3,74) = 1.095d0
dummy(3,75) = 1.096d0
dummy(3,76) = 1.095d0
dummy(3,77) = 1.095d0
dummy(3,78) = 1.096d0
dummy(3,79) = 1.095d0
dummy(3,80) = 1.095d0
dummy(3,81) = 1.095d0
dummy(3,82) = 1.095d0

end
```

```
************************************************************************
subroutine f_def(dummy)
************************************************************************
* This subprogram defines the values of "f" which allow calculation    *
* of viscosity and thermal conductivity to third order.  These values  *
* are found in Table I-P.                                              *
************************************************************************

implicit none
double precision dummy(3,82)
dummy(1,1) = 0.3d0
dummy(1,2) = 0.5d0
dummy(1,3) = 0.75d0
dummy(1,4) = 1.d0
dummy(1,5) = 1.25d0
dummy(1,6) = 1.5d0
dummy(1,7) = 2.d0
dummy(1,8) = 2.5d0
dummy(1,9) = 3.d0
dummy(1,10) = 4.d0
dummy(1,11) = 5.d0
dummy(1,12) = 10.d0
dummy(1,13) = 50.d0
dummy(1,14) = 100.d0
dummy(1,15) = 400.d0

dummy(2,1) = 1.0014d0
dummy(2,2) = 1.0002d0
dummy(2,3) = 1.0000d0
dummy(2,4) = 1.0000d0
dummy(2,5) = 1.0001d0
dummy(2,6) = 1.0004d0
dummy(2,7) = 1.0014d0
dummy(2,8) = 1.0025d0
dummy(2,9) = 1.0034d0
dummy(2,10) = 1.0049d0
dummy(2,11) = 1.0058d0
dummy(2,12) = 1.0075d0
dummy(2,13) = 1.0079d0
dummy(2,14) = 1.0080d0
```

```
dummy(2,15) = 1.0080d0

dummy(3,1) = 1.0022d0
dummy(3,2) = 1.0003d0
dummy(3,3) = 1.0000d0
dummy(3,4) = 1.0001d0
dummy(3,5) = 1.0002d0
dummy(3,6) = 1.0006d0
dummy(3,7) = 1.0021d0
dummy(3,8) = 1.0038d0
dummy(3,9) = 1.0052d0
dummy(3,10) = 1.0076d0
dummy(3,11) = 1.0090d0
dummy(3,12) = 1.0116d0
dummy(3,13) = 1.0124d0
dummy(3,14) = 1.0125d0
dummy(3,15) = 1.0125d0

end
**************************************************************************
subroutine Chooser(dummy, passback, Tstar, column)
**************************************************************************
* This subprogram chooses the correct tabular value given a particular   *
* Tstar.                                                                 *
**************************************************************************

implicit none
integer n, column
double precision dummy(3,82), passback, Tstar, m, b
n=1
do while (dummy(1,n) .le. Tstar)
  n = n + 1
end do
n = n - 1
m = (dummy(column,n+1) - dummy(column,n)) / (dummy(1,n+1) - dummy(1,n))
b = dummy(column,n) - m * dummy(1,n)
passback = m * Tstar + b
end
```

Vita

James Richard Belcher was born on June 24, 1959 in Wytheville, Virginia to Mr. and Mrs. Edgar Earl Belcher. His father retired from the United States Navy in 1969 and moved the family to Arkansas. James attended Oakridge Central High School in Ravenden Springs, Arkansas and graduated in 1977. During high school James enlisted in the United States Navy and was sent to basic training in Orlando, Florida. He was selected for the Naval Electronics Program and completed Electronics, Radar Repair, and Satellite Repair Schools before being assign to Seventh Fleet Communications, where he served aboard the USS Oklahoma City and USS Blue Ridge in the South Pacific. James received Humanitarian Service, Good Conduct and Battle Efficiency medals and was honorably discharged in May 1983. James attended the University of Central Arkansas located in Conway, Arkansas where he received a Bachelor of Science degree (Physics) in 1987. He accepted a position with the Arkansas State Emergency Services until June of 1988 when he was accepted to Graduate School at the University of Mississippi.

James is married to Patricia M. Belcher and they have three children. He is very active in community and youth athletics. He is a volunteer coach for the Oxford Park Commission. He is a member of Sigma Xi, Acoustical Society of America, American Softball Coaching Association, and Oxford Youth Association. James intends to pursue a career in research and/or teaching in the field of Physics and Acoustics. His permanent address is: 159 Walker Road, Conway, AR 72466.